

*Commenced Publication in 1973*

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

## Editorial Board

David Hutchison

*Lancaster University, UK*

Takeo Kanade

*Carnegie Mellon University, Pittsburgh, PA, USA*

Josef Kittler

*University of Surrey, Guildford, UK*

Jon M. Kleinberg

*Cornell University, Ithaca, NY, USA*

Friedemann Mattern

*ETH Zurich, Switzerland*

John C. Mitchell

*Stanford University, CA, USA*

Moni Naor

*Weizmann Institute of Science, Rehovot, Israel*

Oscar Nierstrasz

*University of Bern, Switzerland*

C. Pandu Rangan

*Indian Institute of Technology, Madras, India*

Bernhard Steffen

*University of Dortmund, Germany*

Madhu Sudan

*Massachusetts Institute of Technology, MA, USA*

Demetri Terzopoulos

*University of California, Los Angeles, CA, USA*

Doug Tygar

*University of California, Berkeley, CA, USA*

Moshe Y. Vardi

*Rice University, Houston, TX, USA*

Gerhard Weikum

*Max-Planck Institute of Computer Science, Saarbruecken, Germany*

Opher Etzion Tsvi Kuflik  
Amihai Motro (Eds.)

# Next Generation Information Technologies and Systems

6th International Conference, NGITS 2006  
Kibbutz Shefayim, Israel, July 4-6, 2006  
Proceedings

## Volume Editors

Opher Etzion  
IBM Haifa Labs  
Haifa University Campus, Haifa 31905, Israel  
E-mail: opher@il.ibm.com

Tsvi Kuflik  
University of Haifa  
MIS Department  
Mount Carmel, Haifa, 31905, Israel  
E-mail: tsviak@is.haifa.ac.il

Amihai Motro  
George Mason University  
ISE Department  
Fairfax, VA 22015, USA  
E-mail: ami@gmu.edu

Library of Congress Control Number: 2006928068

CR Subject Classification (1998): H.4, H.3, H.5, H.2, D.2.12, C.2.4

LNCS Sublibrary: SL 3 – Information Systems and Application, incl. Internet/Web and HCI

|         |   |
|---------|---|
| ISSN    | 0302-9743   |
| ISBN-10 | 3-540-35472-7 Springer Berlin Heidelberg New York     |
| ISBN-13 | 978-3-540-35472-7 Springer Berlin Heidelberg New York |

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media  
springer.com

© Springer-Verlag Berlin Heidelberg 2006  
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India  
Printed on acid-free paper SPIN: 11780991 06/3142 5 4 3 2 1 0

# Preface

Information technology is a rapidly changing field in which researchers and developers must continuously set their vision on the next generation of technologies and the systems that they enable. Suitably, Next-Generation Information Technologies and Systems (NGITS) is an ongoing series of workshops that provides a forum for presenting and discussing the latest advances in information technology. NGITS took are international events held in Israel; previous workshops took place in 1993, 1995, 1997, 1999 and 2002.

The call for papers for the 2006 workshop was answered by 138 papers on a very diverse range of subjects, thus presenting a considerable challenge to the Program Committee. Each of these papers received professional reviewing from at least three experts, and eventually 32 papers (less than 25%) were selected for presentation at the workshop and inclusion in these proceedings. Of these, 28 are full-length papers and 4 are short papers. In addition, several research works were selected to display “poster” presentations, and a number of research projects exhibited demonstrations. Finally, the workshop featured three keynote lectures by notable experts. The selected papers may be classified roughly in ten broad areas:

- Information systems development
- Distributed systems
- Semi-structured data
- Data mining and agent-oriented computing
- User-oriented design
- Frameworks, models and taxonomies
- Simulation and incremental computing
- Information integration
- Security and privacy
- Next-generation applications

This event is the culmination of efforts by many talented and dedicated individuals. We are pleased to extend our thanks to the authors of all submitted papers, the members of the Steering Committee, the members of the Program Committee, and the external reviewers. Special thanks are due to Dan Berry, for his discrete handling of papers that presented conflicts of interest. Many thanks are also due to Nilly Schnapp for local organization and logistics, and to Lior Shmueli for managing the website and all other things technical. Finally, we are pleased to acknowledge the support of our institutional sponsors: The Faculty of Social Sciences and the MIS Department at the University of Haifa, and the IBM Research Lab in Haifa.

July 2006

Tsvi Kuflik  
General Chair  
Opher Etzion and Amihai Motro  
Program Co-Chairs



# Organization

## General Chair

Tsvi Kuflik

## Steering Committee

|               |             |                  |
|---------------|-------------|------------------|
| Opher Etzion  | Avigdor Gal | Alon Halevy      |
| Amihai Motro  | Ron Pinter  | Avi Silberschatz |
| Peretz Shoval | Shalom Tsur |                  |

## Program Committee Chairs

|              |              |
|--------------|--------------|
| Opher Etzion | Amihai Motro |
|--------------|--------------|

## Program Committee Vice Chair

Dan Berry

## Program Committee

|                     |                    |                       |
|---------------------|--------------------|-----------------------|
| Serge Abiteboul     | Nabil Adam         | Hamideh Afsarmanesh   |
| Eugene Agichtein    | Irit Askira-Gelman | Catriel Beeri         |
| Iris Berger         | Elisa Bertino      | Martin Bichler        |
| Andrei Broder       | Jen-Yao Chung      | Alessandro D'Atri     |
| Vijay Dialani       | AnHai Doan         | Asuman Dogac          |
| Carlotta Domeniconi | Ophir Frieder      | Paolo Giorgini        |
| Ehud Gudes          | Alfons Kemper      | Larry Kerschberg      |
| David Konopnicki    | Manolis Koubarakis | Yossi Matias          |
| Tova Milo           | Mukesh Mohania     | Felix Naumann         |
| George Papadopoulos | Norman Paton       | Francesco Ricci       |
| Naphtali Rishe      | Doron Rotem        | Steve Schach          |
| Yuval Shahar        | Oded Shmueli       | Charles A. Shoniregun |
| Pnina Soffer        | Carlo Strapparava  | Bernhard Thalheim     |
| Yair Wand           | Ouri Wolfson       | Carlo Zaniolo         |

## **Additional Reviewers**

|                     |                     |                       |
|---------------------|---------------------|-----------------------|
| Aybar C. Acar       | Benni Arazi         | Yijian Bai            |
| Khalid Belhajjame   | Shlomo Berkovsky    | Alexander Bilke       |
| Jens Bleiholder     | Vlaidmir Braverman  | Neslihan Bulut        |
| Nunzio Casalino     | Hu Cao              | Bogdan Cautis         |
| Dario Cavada        | Kuo-Ming Chao       | Mariangela            |
| Contenti            | Bonaventura Coppola | Luiz Marcio Cysneiros |
| Marco De Marco      | Pedro DeRose        | Chrysanne DiMarco     |
| Nurit Galoz         | Ozgur Gulderen      | Jeff Heard            |
| Yildiray Kabak      | Zoi Kaoudi          | Gokce Banu Laleci     |
| Yoonkyong Lee       | Yinsheng Li         | Jessica Lin           |
| Juhong Liu          | Sean Luke           | Steven Lynden         |
| Robert McCann       | Iris Miliaraki      | Oleg Missikoff        |
| Simon Msanjila      | Tuncay Namli        | Alper Okcan           |
| Mehmet Olduz        | Frank Olken         | Jeffery Orchard       |
| Umut Orhan          | Ekow Otoo           | Christoforos          |
| Michael Pantazoglou | Nguyen Nhat Quang   | Andrea Resca          |
| Boris Rozenberg     | Mayssam Sayyadian   | Tayfun Sen            |
| Pierre Senallart    | Warren Shen         | Paolo Spagnoletti     |
| Arnon Sturm         | Ibrahim Tasyurt     | Hetal Thakkar         |
| Aimilia Tzanavar    | Ozgul Unal          | Marcos Vieira         |
| Ronen Waisenberg    | Wensheng Wu         | Bo Xu                 |
| Bojun Yan           | Huabei Yin          | Mustafa Yuksel        |
| Xin Zhou            |                     |                       |

## **Local Arrangements Chair**

Nilly Schnapp

## **Website Manager**

Lior Shmueli

# Organization

## General Chair

Tsvi Kuflik

## Steering Committee

Opher Etzion  
Amihai Motro  
Peretz Shoval

Avigdor Gal  
Ron Pinter  
Shalom Tsur

Alon Halevy  
Avi Silberschatz

## Program Committee Chairs

Opher Etzion      Amihai Motro

## Program Committee Vice Chair

Dan Berry

## Program Committee

Serge Abiteboul  
Eugene Agichtein  
Iris Berger  
Andrei Broder  
Vijay Dialani  
Carlotta Domeniconi  
Ehud Gudes  
David Konopnicki  
Tova Milo  
George Papadopoulos  
Naphtali Rishe  
Yuval Shahar  
Pnina Soffer  
Yair Wand

Nabil Adam  
Irit Askira-Gelman  
Elisa Bertino  
Jen-Yao Chung  
AnHai Doan  
Ophir Frieder  
Alfons Kemper  
Manolis Koubarakis  
Mukesh Mohania  
Norman Paton  
Doron Rotem  
Oded Shmueli  
Carlo Strapparava  
Ouri Wolfson

Hamideh Afsarmanesh  
Catriel Beeri  
Martin Bichler  
Alessandro D'Atri  
Asuman Dogac  
Paolo Giorgini  
Larry Kerschberg  
Yossi Matias  
Felix Naumann  
Francesco Ricci  
Steve Schach  
Charles A. Shoniregun  
Bernhard Thalheim  
Carlo Zaniolo

## **Additional Reviewers**

|                     |                     |                       |
|---------------------|---------------------|-----------------------|
| Aybar C. Acar       | Benni Arazi         | Yijian Bai            |
| Khalid Belhajjame   | Shlomo Berkovsky    | Alexander Bilke       |
| Jens Bleiholder     | Vlaidmir Braverman  | Neslihan Bulut        |
| Nunzio Casalino     | Hu Cao              | Bogdan Cautis         |
| Dario Cavada        | Kuo-Ming Chao       | Mariangela            |
| Contenti            | Bonaventura Coppola | Luiz Marcio Cysneiros |
| Marco De Marco      | Pedro DeRose        | Chrysanne DiMarco     |
| Nurit Galoz         | Ozgur Gulderen      | Jeff Heard            |
| Yildiray Kabak      | Zoi Kaoudi          | Gokce Banu Laleci     |
| Yoonkyong Lee       | Yinsheng Li         | Jessica Lin           |
| Juhong Liu          | Sean Luke           | Steven Lynden         |
| Robert McCann       | Iris Miliaraki      | Oleg Missikoff        |
| Simon Msanjila      | Tuncay Namli        | Alper Okcan           |
| Mehmet Olduz        | Frank Olken         | Jeffery Orchard       |
| Umut Orhan          | Ekow Otoo           | Christoforos          |
| Michael Pantazoglou | Nguyen Nhat Quang   | Andrea Resca          |
| Boris Rozenberg     | Mayssam Sayyadian   | Tayfun Sen            |
| Pierre Senallart    | Warren Shen         | Paolo Spagnoletti     |
| Arnon Sturm         | Ibrahim Tasyurt     | Hetal Thakkar         |
| Aimilia Tzanavar    | Ozgul Unal          | Marcos Vieira         |
| Ronen Waisenberg    | Wensheng Wu         | Bo Xu                 |
| Bojun Yan           | Huabei Yin          | Mustafa Yuksel        |
| Xin Zhou            |                     |                       |

## **Local Arrangements Chair**

Nilly Schnapp

## **Website Manager**

Lior Shmueli

# Table of Contents

## Full Papers

### Information Integration

|  |    |
|--|----|
| Efficiently Updating Cost Repository Values for Query Optimization<br>on Web Data Sources in a Mediator/Wrapper Environment<br><i>Justo Hidalgo, Alberto Pan, Manuel Álvarez, Jaime Guerrero . . . . .</i> | 1  |
| TupleRank: Ranking Discovered Content in Virtual Databases<br><i>Jacob Berlin, Amihai Motro . . . . .</i>  | 13 |

### Next-Generations Applications

|  |    |
|--|----|
| Analysis of Queries Reaching SHIL on the Web - An Information<br>System Providing Citizen Information<br><i>Gilad Ravid, Judit Bar-Ilan, Sheizaf Rafaeli,<br/>Shifra Baruchson-Arbib . . . . .</i> | 26 |
| On Mediated Search of the United States Holocaust Memorial Museum<br>Data<br><i>Jefferson Heard, Jordan Wilberding, Gideon Frieder, Ophir Frieder,<br/>David Grossman, Larry Kane . . . . .</i>    | 38 |
| A Repository of Services for the Government to Businesses Relationship<br><i>Daniele Barone, Gianluigi Viscusi, Carlo Batini, Paolo Naggari . . . . .</i>  | 47 |

### Information Systems Development

|   |    |
|---|----|
| Towards Automatic Integration of Persistency Requirements<br>in Enterprise-Systems – The Persistent-to-Persistent Patterns<br><i>Mira Balaban, Lior Limonad . . . . .</i>                 | 59 |
| Consistency of UML Class Diagrams with Hierarchy Constraints<br><i>Mira Balaban, Azzam Marace . . . . .</i>   | 71 |
| A Framework-Based Design for Supporting Availability of Layered<br>Distributed Applications<br><i>Heung Seok Chae, Jaegeol Park, Jinwook Park, Sungho Ha,<br/>Joon-Sang Lee . . . . .</i> | 83 |

## Security and Privacy

|   |     |
|---|-----|
| Web Application Security Gateway with Java Non-blocking IO<br><i>Zhenxing Luo, Nuermaitaiti Heilili, Dawei Xu, Chen Zhao,<br/>Zuoquan Lin</i> .....           | 96  |
| Microaggregation for Database and Location Privacy<br><i>Josep Domingo-Ferrer</i> .....   | 106 |
| Efficient Caching Strategies for Gnutella-Like Systems to Achieve<br>Anonymity in Unstructured P2P File Sharing<br><i>Byung Ryong Kim, Ki Chang Kim</i> ..... | 117 |

## Semi-structured Data

|   |     |
|---|-----|
| Conjunctive Queries over DAGs<br><i>Royi Ronen, Oded Shmueli</i> .....  | 129 |
| Incrementally Computing Ordered Answers of Acyclic Conjunctive<br>Queries<br><i>Benny Kimelfeld, Yehoshua Sagiv</i> ..... | 141 |
| Count-Constraints for Generating XML<br><i>Sara Cohen</i> .....   | 153 |

## Frameworks, Models and Taxonomies

|   |     |
|---|-----|
| A Data Model for Music Information Retrieval<br><i>Tamar Berman</i> .....   | 165 |
| A Taxonomy and Representation of Sources of Uncertainty in Active<br>Systems<br><i>Segev Wasserkrug, Avigdor Gal, Opher Etzion</i> .....        | 174 |
| Analyzing Object-Oriented Design Patterns from an Object-Process<br>Viewpoint<br><i>Galia Shlezinger, Iris Reinhartz-Berger, Dov Dori</i> ..... | 186 |

## Simulation and Incremental Computing

|  |     |
|--|-----|
| Modeling and Simulation of Grid Information Service<br><i>Xia Xie, Hai Jin, Jin Huang, Qin Zhang</i> ..... | 198 |
|--|-----|

|  |     |
|--|-----|
| Simulations of Distributed Service-Based Content Adaptation for Network Optimization<br><i>Eric Y. Cheng, Shuo Hung Jian</i> ..... | 210 |
|--|-----|

## Distributed Systems

|  |     |
|--|-----|
| Second Order Snapshot-Log Relations: Supporting Multi-directional Database Replication Using Asynchronous Snapshot Replication<br><i>Yochai Ben-Chaim, Avigdor Gal</i> ..... | 221 |
| The Replica Management for Wide-Area Distributed Computing Environments<br><i>Jaechun No, Chang Won Park, Sung Soon Park</i> .....   | 237 |
| Developing Parallel Cell-Based Filtering Scheme Under Shared-Nothing Cluster-Based Architecture<br><i>Jae-Woo Chang, Tae-Woong Wang</i> .....                                | 249 |

## User Oriented Design

|  |     |
|--|-----|
| How Deep Should It Be? On the Optimality of Hierarchical Architectures<br><i>Amihai Motro, Alessandro D'Atri, Eli Gafni</i> .....                | 260 |
| A Spreadsheet Client for Web Applications<br><i>Dirk Draheim, Peter Thiemann, Gerald Weber</i> .....   | 274 |
| Dynamic Construction of User Defined Virtual Cubes<br><i>Dehui Zhang, Shaohua Tan, Dongqing Yang, Shiwei Tang, Xiuli Ma, Lizheng Jiang</i> ..... | 287 |

## Data Mining and Agent-Oriented Computing

|   |     |
|---|-----|
| Automatic Discovery of Regular Expression Patterns Representing Negated Findings in Medical Narrative Reports<br><i>Roni Romano, Lior Rokach, Oded Maimon</i> ..... | 300 |
| A Hybrid Method for Speeding SVM Training<br><i>Zhi-Qiang Zeng, Ji Gao, Hang Guo</i> .....  | 312 |
| Design of Service Management System in OSGi Based Ubiquitous Computing Environment<br><i>Seungkeun Lee, Jeonghyun Lee</i> .....                                     | 321 |

## Short Papers

|  |     |
|--|-----|
| Creating Consistent Diagnoses List for Developmental Disorders Using UMLS<br><i>Nuaman Asbeh, Mor Peleg, Mitchell Schertz, Tsvi Kuflik</i> . . . . . | 333 |
| Enhancing Domain Engineering with Aspect-Orientation<br><i>Iris Reinhartz-Berger, Alex Gold</i> . . . . .  | 337 |
| An Agent-Oriented Approach to Semantic Web Services<br><i>In-Cheol Kim</i> . . . . .   | 341 |
| Biometrics Authenticated Key Agreement Scheme<br><i>Eun-Jun Yoon, Kee-Young Yoo</i> . . . . .  | 345 |

## Posters and Demonstrations

|   |     |
|---|-----|
| The Third Query Paradigm<br><i>Mikhail Gilula</i> . . . . .   | 350 |
| $\tau$ -xSynopsis – a System for Run-Time Management of XML Synopses<br><i>Natasha Drukh, Yariv Matia, Yossi Matias, Leon Portman</i> . . . . .         | 351 |
| LTS: The List-Traversal Synopses System<br><i>Michael Furman, Yossi Matias, Ely Porat</i> . . . . .   | 353 |
| The Promise and Challenge of Multidimensional Visualization<br><i>Alfred Inselberg</i> . . . . .  | 355 |
| <i>OPOSSUM</i> : Bridging the Gap Between Web Services and the Semantic Web<br><i>Eran Toch, Iris Reinhartz-Berger, Avigdor Gal, Dov Dori</i> . . . . . | 357 |
| <i>ProMo</i> - A Scalable and Efficient Framework for Online Data Delivery<br><i>Haggai Roitman, Avigdor Gal, Louiqa Raschid</i> . . . . .              | 359 |

## Invited Talks

|  |     |
|--|-----|
| Next Generation Enterprise IT Systems<br><i>Donald F. Ferguson</i> . . . . .                                 | 361 |
| The Future of Web Search: From Information Retrieval to Information Supply<br><i>Andrei Broder</i> . . . . . | 362 |



Is There Life After the Internet?  
    *Yechiam Yemini* ..... 363

**Author Index** ..... 365

# Efficiently Updating Cost Repository Values for Query Optimization on Web Data Sources in a Mediator/Wrapper Environment

Justo Hidalgo<sup>1</sup>, Alberto Pan<sup>2,\*</sup>, Manuel Álvarez<sup>2</sup>, and Jaime Guerrero<sup>3</sup>

<sup>1</sup> Denodo Technologies, Inc.

Madrid, Spain

jhidalgo@denodo.com

<sup>2</sup> Department of Information and Communications Technologies

University of A Coruña, Spain

{apan, mad}@udc.es

<sup>3</sup> jaimeguerrero@wanadoo.es

**Abstract.** Optimizing accesses to sources in a mediator/wrapper environment is a critical need. Due to a variety of reasons, relational-based optimization techniques are of no use when having to handle HTTP-based web sources, so new approaches which take into account client/server communication costs must be devised. This paper describes a cost model that stores values from a complete set of web source-focused parameters obtained by the web wrappers, by using a novel updating technique that handles the values measured by the wrappers in previous query executions, and generates a new model instance in each new iteration with an efficient processing cost. This instance allows rapid value updates caused by changes of the server quality or bandwidth, so typical in this context. The results of these techniques are demonstrated both theoretically and by means of an implementation showing how performance improves in real-world web sources when compared to classical approaches.

## 1 Introduction

Virtual Databases, also called Mediators [14], allow the obtaining, unification and sampling of data residing in heterogeneous environments, as much because they are stored in different repositories, like their geographic location. Virtual Databases differ from standard databases because data are not really stored in the database, but remotely in several heterogeneous sources. Mediators form a middleware layer between the user and the sources to be integrated, providing a single point of access or data access layer to the underlying data. Each data source exposes its search capabilities through the set of parameters which can be used when querying it, along with the possible values and constraints for each parameter. At the time of executing a query in a mediator, a set of query plans is generated, each of which retains the same capabilities and restrictions, thus being able to respond to the query correctly, but with each of

---

\* Alberto Pan's work was partially supported by the "Ramón y Cajal" programme of the Spanish Ministry of Education and Science.

them having different search methods of concrete sources, different sources, operator execution strategies, and so on. For example, imagine the case shown in Fig. 1, in which there are two electronic bookshop sources from which to search for works written by a particular author. Besides, there is a complementary set of web sources which store critics' reviews from books. We are interested on being able to query the system in order to retrieve all works written by an author, along with reviews made to that work from one of the sources. The basic query plan to be performed is therefore a relational algebra union operation between shop A and shop B, and a join between that union and the review site. But there can be different query plans depending, for example, in the type of join strategy to be performed, a hash join or a nested-loop join. Besides, the system could choose one review site or other depending on their reliability in a particular moment.

The diagram shows three web forms arranged horizontally. The first form is labeled 'SHOP A' and contains two input fields: 'TITLE' and 'AUTHOR', each followed by an asterisk (\*). Below these fields is a green button labeled 'SUBMIT'. The second form is labeled 'SHOP B' and contains the same 'TITLE' and 'AUTHOR' fields with asterisks and a 'SUBMIT' button. The third form is labeled 'REVIEW' and contains the same 'TITLE' and 'AUTHOR' fields with asterisks, plus two additional fields: 'REVIEW' and 'SCORE', each followed by an asterisk. Below these four fields is a green 'SUBMIT' button. Below the three forms, centered, is the text '\*: ONE OR THE OTHER'.

**Fig. 1.** Electronic Bookshop Example

In a specific moment of time, only one of those plans will be optimal, that is to say, it will be the one that allows a faster and/or reliable execution, optimizing the resources to use. The importance of improving response times in web environments, where quality of service is not assured, has impelled this line of research in the last years.

The first approaches to query optimization were performed in the field of multidatabases [8], where cost estimation in relational databases uses statistics of the database to compute the cost of each operator in each plan. The authors propose a combination of a generic cost model with specific information of each wrapper. This concept is valuable when the information is very homogeneous. However, this approach is not directly applicable to virtual databases in the web environment, had mainly to the following reasons: (1) sources do not usually offer statistical information, and (2) costs of communications are not determined easily and can vary. Besides, remote communication costs are not considered as part of the model, which makes this idea nonviable in a web environment due to the lack of quality of service in the TCP/IP protocol which makes response times impossible to manage [15].

[9] treats this subject more extensively regarding the required group of parameters when taking into account communication costs among agents and remote sources. Nevertheless, cost propagation formulae provided are not complete enough for a web data source-enabled, industrial system.

Other investigations have offered different options. [3] focuses on coefficient estimation in a generic cost model; this approach is not useful enough with web data sources, since it relies on the remote databases to provide cost statistics.

A more interesting solution for our case is the one proposed in the Hermes project, specified in [1]; the cost model is evaluated from historical information, which can be used for later estimation for each execution plan cost. Hermes concept has been improved by other works such as [16], where a two-phase optimizer is depicted, or [7], where its research on dynamic adaptiveness starts from retrieved cost information.

Nonetheless, little research has been done neither on what kind of cost parameters should be stored in the cost repository from the wrappers when web sources are involved, nor on how these values retrieved in each measurement should be processed so as to better infer how the source will behave in the near future, and in a feasible way in terms of efficiency. Work by Rahal et al. [12], focused on multidatabases, propose a couple of approaches so that a cost model is able to get adjusted when the costs change; their first approach is interesting from a mediator point of view (the second one requires that changes are not abrupt so there is no need of a new model for each value measurement, which, in the case of web data sources characterization, can not be confirmed at all), even though cost information is obtained directly from the repositories (currently imposible to achieve in a web environment). Rahal defines a cost model as the independently-weighted sum of the characterization parameters, and describes a series of techniques to adjust them when the database behaviour changes. This is achieved by using an initial cost model,  $M_0$ , which is updated after each sample query with an asymptotically tight bound of  $\theta(n^2)$ , where  $n$  is the number of parameters. Our criterion is that this approach is not the best when cost information can not be obtained directly from the database, such as in web sources, so a new model is shown with a simpler statistical method and is demonstrated how the update can be achieved with a bound of  $\theta(1)$  on the number of sample queries.

The rest of the paper is organized as follows. Section 2 introduces the mediator/wrapper environment and the reasons why web sources optimization must be managed differently. Section 3 describes the cost repository structure and defines the model parameters. The fourth section defines how to obtain the best query plan by using this repository, and an efficient technique to update every parameter value in each query execution is defined and demonstrated in section 5. Section 6 shows our experimental results obtained by an implementation of the techniques described above and section 7 summarizes on conclusions.

## 2 Analysis of the Problem

Once received the query, the Mediator's Plan Generator produces the different possible plans of execution. Each plan will be made up of a set of subqueries over a set of remote sources; each one of these sources will return back a series of results that, after postprocessing and unification, allow to obtain the final result. The Plan Generator can generate a great amount of possible query plans for a single global user query; therefore, obtaining the optimal query plan for this specific moment is a task of the Logical Optimizer.

This paper will be facing the cost model for logical optimization, taking into account the cost information provided by the wrappers when accessing the web sources, and how these data are stored in the cost repository. Other issues must be taken into

account in mediator optimization such as physical optimization, as described in previous works [5][6][9][11][16].

The factor that influences most decisively in the response time is the transmission of information through the network. In particular, web sources present the following fundamental problems:

- in sources with session data it is usually necessary to make several page requests (navigational sequences) to execute the query in a complete fashion. Thus, the session initialization cost can be as high as (or higher than) the cost of accessing the result pages. Besides, this initialization cost is not constant; in a web source that requires browsing three pages before accessing the query form page, the initialization time will be lower if a browser is already positioned on the second page.
- information stored in a web source for a single item (e.g., a book) can be dispersed along a set of connected pages (such as detail pages). Nonetheless, the user may require these tokens to be shown in an aggregated manner.
- it can also be necessary to browse through several pages in order to obtain all the results of a query (in web sites where answers are returned by intervals).
- an adequate Quality of Service level cannot be assured in any transmission through the web since the underlying TCP/IP communication protocol does not support it.

Our proposal is based on two main improvements over current state of the art on mediator optimization. Firstly, a cost-based model will be developed, taking special attention to the parameters specially defined for web sources; the degree of definition of these parameters will allow to better characterize the query cost propagation. Secondly, a novel technique for updating the cost repository values will be shown and demonstrated, which will improve current techniques due to both the quality of the values inferred (taking into account web source behaviours) and its efficiency.

### 3 Cost Repository Structure

A cost repository is the element which stores that information so that the modules of physical and logical optimization use them easily. Typically, a cost repository follows the structure described in [1], that uses reductions by means of lossy and lossless tables. The statistical data of each query launched by a wrapper are stored: thus, the system can access the atomic elements which are grouped by means of different statistical formulae. The approach described and demonstrated in this paper has as an advantage that detail information from each query execution needs not be stored; the repository only keeps the average parameter values (as processed by the technique shown in the following two sections) plus a few internal values from the previous query execution, enough to update the model in each new query execution.

#### 3.1 Cost Repository Parameters

The basic components of the cost repository will be explained in the following paragraphs:

- *Initialization time*: time required for the browser to navigate from the initial URL to the query page. It includes any authentication and information-filling form and/or

navigation required to arrive to the query form. Even though this parameter could include more sub-parameters (p.e. connection time, authentication time, ...), these are not actually needed to choose among different physical access paths.

- *First Result List Time*: time required for the browser to navigate from the search form/link to the first page of results. It is important to realize that obtaining each result might imply browsing a set of one or more detail pages.
- *Following Results Time*: many web applications show their results in different consecutive pages. Even though a web source can have a very fast response time for the main page, or even a rapid access to the first page of results, it might not be so optimized for the rest of results. Thus, a web source which could be ranked in the first places of our system if the query to be executed returns only a few results, could nevertheless be a slower one when the results to be retrieved are many. This parameter stores the total time involved in obtaining the whole list of result pages, from the second to the last one.
- *Number of Tuples in the First Page* (Máximo Number of Tuples per Page): this parameter allows the system to have information about how many results are usually retrieved in the first result page. This is used in cost propagation.
- *Total Number of Tuples*: this parameter keeps the whole number of tuples that a query execution returns. This is as well used in cost propagation for some operators, such as the nested-loop join one.
- *Size of First Page of Results*: stores the size of the first page of a query result.

The repository differentiates between first result list time and following results time because of two reasons: firstly, in many sources the query form is not the same as the following result form; secondly, the study of how web sources behave tells us that it is typical that processing the initial query takes less than processing the following intervals. This second observation allows the user to decide whether to prioritize query plans which sources return the first list of results in a faster way, or those which return the whole list of results faster. This decision affects the parameters to be used in the query plan cost propagation (for example, “number of tuples in the first page” or “total number of tuples” will be used depending on the choice).

An important characteristic in the cost repository is the storage of information by each search method and time section. In our preliminary studies we have observed great existing differences of web source performance in different time sections. The choice between a search method or another one also depends on the time section in which the query is made.

Finally, information about failed connection percentage will allow the mediator to be able to choose between fast sources – though inclined to fail-, and slower but trustworthier ones.

These parameters will be updated every time a query is executed and their cost values are obtained by the wrapper, so that they can be used in the bottom-up query plan cost propagation in order to have a single value for each candidate plan, so to choose the optimal one. In industrial mediator systems, the set of potential query plans which are generated for a single view can be exponentially related to the number of search methods that the query makes use of. Obviously, this is unacceptable when the cost of all of them must be obtained by propagating the cost from the leaves (base views or search methods) to the root node. A solution proposed, among others by [16], divides the choice of possible query plans in two phases: compilation time,

when the views are being created (i.e. transformed into query plans) and at execution time, when a query is executed.

### 3.2 Use of Cache at Compilation and Execution Time

At compilation time, we do not have any type of current information about the system; only historical data from a cost repository can be used. At execution time, when a query arrives, the system will automatically be able to determine whether the values of any of its underlying search methods can be completely replied by cache; in these cases, their cost value will be 0. In the case of cache at compilation time, the system can obtain the “probability that cache is used” for a particular search method. This will be used to leverage the cost of that search method according to this datum: if a search method has a high cost according to the logical type cost repository, but a very high probability that its values will be stored at cache, it could be feasible to use it in a query plan.

At last, the parameters described in this paper are focused on the optimization of the mediator logical layer, but the cost repository also stores data useful for physical layer optimization, which is not explained here due to lack of space [5].

## 4 Query Plan Optimization Processing

Once the wrapper returns the cost information from a query performed over a particular source, the optimization system must update the repository so that, when the Plan Generator creates the candidate query plans, it can invoke the Logical Optimizer in order to propagate the current cost parameters from the base views up to the root node of each plan so to choose the optimal one.

The updating technique proposed in this work will be used as many times as there are parameters in the cost model. The updated values are required by the bottom-up cost propagation algorithm, thus allowing different strategies according to user or system configuration. For example, we can define the cost of a nested-loop join. This strategy requires the mediator to query the first source, obtain the number of distinct values for the join attributes and, for each unique combination of these values, query the second source and then join the results. In a mediator environment, the number of distinct values of the first source can not be previously determined, so it is computed as the `TotalNumberOfTuples` parameter multiplied by a factor  $N$  which has been empirically defined as  $1/10$ , following the classical relational selectivity default value. Thus, the final formula reads as follows:

Let  $\text{tupleNumber}(R_i)$  = number of tuples of relation  $i$ , parameter retrieved from the cost repository.

So we define:

$$\text{cost}(\text{JOIN}_N(R_1, R_2)) = \text{time}(R_1) + \text{ceiling}(N \cdot \text{tupleNumber}(R_1)) \cdot \text{time}(R_2). \quad (1)$$

A nested-loop join strategy can be useful in web sources whenever query capabilities do not allow a hash join (because join attributes can not be directly used to query the second source) or when the time required to access the “following results” pages from the second source takes much more than expected.

## 5 The Aggregated Time Method

The updating method is defined as follows: let us consider a list of pairs  $(t_i, v_i)$ ,  $i=1..k$ , where  $t_i$  are the time values which are not equally distributed over time, and  $v_i$  are the values of certain magnitude, measured over such time values. These are the values obtained by the wrapper from a specific cost parameter, over a period of time. The list is incremented with each new measurement  $v_{k+1}$  in  $t_{k+1}$ .

Our goal is to generate a weighted average of  $(v_i)$ ,  $M(k) = \sum \eta_i(k) v_i$ ,  $i=1..k$ , in time  $t_k$ , so that: (1) the weights of the most recent values are “more important” than the ones of the older values; (2) the weights of the most recent values are much more important if the other ones are much older. The small difference between this and the previous requirement is critical in our model; what it says is that, since the web environment does not have an accurate quality of service characterization, older values of the parameter are still taken into account but only as a fraction of what they would be; this way, changes of quality of service in the web source are rapidly incorporated into the model used for cost propagation (i.e. the statistical expected value of the parameter changes its value faster than in a standard average distribution function); and, (3) the calculation of  $M(k+1)$  is performed in an efficient way.

The probability function necessary for the cost model must follow that: (1)  $\sum \eta_i(k) = 1$ ,  $i=1..k$ ; (2) this factor strengthens the decrease of the value with regard to time (the older the value was measured, the less important its value will be, tending to zero); and (3) also desirable, this decrease must be outliers-free: those values that are  $n$  times the current standard deviation will be deleted. It could also be achieved by creating a threshold defined by a percentage ceiling over and under the current average value obtained by the model. Our approach uses the last choice in order to avoid the storage of detail value information for every query execution.

With this information in mind, our proposal is the following: the choice of the  $\eta_i(k)$  is subjective, and depends on how we want to show the “importance” of the last data element with respect to the rest. The choice of the following  $\eta_i(k)$  verifies the previous goals. Let  $T_k = \sum t_i$ ,  $i=1..k$  (1) be the accumulated time until datum  $k$ , and  $\eta_i(k) = t_i / T_k$  be the weight of datum “ $i$ ” in time  $k$ . For calculation issues, let us always store the last value  $T_k$ . When a new item is produced:  $(t_{k+1}, v_{k+1})$ , we generate

$$T_{k+1} = T_k + t_{k+1} . \quad (2)$$

Let us now define

$$\lambda_{k+1} = T_k / T_{k+1} . \quad (3)$$

Please notice that this value is the same for every “ $i$ ”. If  $i=1..k$ , then, from (2) and (3),  $\eta_i(k+1) = t_i / T_{k+1} = T_k / T_{k+1} \cdot t_i / T_k = \lambda_{k+1} \eta_i(k)$ , that is:

$$\eta_i(k+1) = \lambda_{k+1} \eta_i(k) . \quad (4)$$

This way, the first “ $k$ ” weights of the new model in  $t_{k+1}$  are obtained by multiplying by  $k+1$  (which is just a number) the last weight  $\eta_{k+1}(k+1) = t_{k+1}/T_{k+1}$ , directly. But since:

$$M(k+1) = \sum \eta_i(k+1) v_i, \quad i=1..k+1,$$

$$M(k+1) = \sum \eta_i(k+1) v_i, \quad i=1..k + \eta_{k+1}(k+1) \cdot v_{k+1}$$



From (4),  $M(k+1) = \sum \lambda_{k+1} \eta_i(k) v_i$ ,  $i=1..k + \eta_{k+1}(k+1) \cdot v_{k+1}$   
 $M(k+1) = \lambda_{k+1} M(k) + \eta_{k+1}(k+1) \cdot v_{k+1}$

$$M(k+1) = \lambda_{k+1} M(k) + \eta_{k+1}(k+1) \cdot v_{k+1} . \quad (5)$$

Thus, obtaining  $M(k+1)$  implies attaining the current and previous values of  $T$ , the previous value of the model  $M$  and the currently retrieved value  $v$ , all of which are independent of the number of stored values  $v_i$  in the cost repository. Thus, the formula has a complexity of  $\theta(1)$  for every new  $v_{k+1}$  in  $t_{k+1}$ . This has very important implications on the cost model proposed in this paper. First of all, the new values can be generated without having to rebuild them from previous ones: we require just a couple of parameters stored from the previous iteration: the new model is independent of the number of values already obtained. Secondly, we are accomplishing the initial constraints, since the model prioritizes the parameter values most recently retrieved by the wrappers; the complexity improves the one of a mean average ( $M(k) = \sum \eta_i(k) v_i$ ,  $i=1..k$ , where  $\eta_i(k) = 1/k$ ) both quantitatively, since this would take  $\theta(k)$  to be computed in each iteration, and qualitatively, because this average does not take into account any of the advantages described above.

Our formula can be generalized by adding the location parameter to it, which allows to take a  $t_0$  different from 0 (which, in practical terms will be usual, since a typical way of processing  $t_i$  is by using the number of milliseconds from January 1<sup>st</sup>, 1970 until that exact time). The formula is basically the same, just by defining:

$t'_i = t_i - t_{\text{start}}$ , where  $t_{\text{start}}$  is the time of the first iteration to be taken into account, which leads us to:

$$M(k+1) = \lambda'_{k+1} M(k) + \eta'_{k+1}(k+1) v_{k+1} . \quad (6)$$

## 6 Experimental Results

An implementation of the cost repository, and update and propagation techniques has been developed on top of a commercially-available mediator middleware from Denodo Technologies, Virtual DataPort [10][13]. A set of experiments has been performed, by executing several queries on a set of different web sources throughout different periods of time from November 2005 to January 2006. The mediator was deployed in a distributed environment running under Windows XP on Pentium 4 PCs, 3.4 GHz with 1 GB of RAM and a 1 MB ADSL connection. Sources have been selected so that they represent the most typical ones that can be found in the Deep Web (transactional web sites which require authentication and form filling before returning a list of results, with or without detail pages, and with or without pages with more results). Table 1 shows some of the web sources which took part in this experimentation, and their basic navigational sequence in NSEQL (Navigational Sequence Language) [13]; this language is used to define sequences of events on the interface provided by the web browser used by the web wrapper. The NSEQL programs in the table allow the wrapper to access the first web page of results. The complete specification of each web wrapper is not shown here for the sake of brevity.

Google Scholar [4] is a very simple source in terms of structure: the first page exposes a form from which to query the application. A list of pages is then retrieved by

Table 1. Web Sources used in the Experiments

| Source                            | Basic NSEQL  |
|-----------------------------------|--|
| Google Scholar (from Google home) | Navigate(http://www.google.com,0); WaitPages(1); ClickOnAnchorByText(more >>, 0, true, 0); WaitPages(1); ClickOnAnchorByText(Scholar, 0, true, 0); WaitPages(1); FindFormByName(f,0); SetInputValue(q,0,@SEARCH); ClickOnElement(btnG,input,0); WaitPages(1);  |
| Citeseer                          | Navigate(http://citeseer.ist.psu.edu/,0); WaitPages(1); FindFormByName(mainform,0); SetInputValue(q,0,@SEARCH); ClickOnElement(submit,INPUT,0); WaitPages(1);  |
| Citation Local Server             | Navigate(http://localhost:8080/citations/login.jsp,0);WaitPages(1); FindFormByName(login,0); SetInputValue(user,0,admin); SetInputValue(password,0,admin); ClickOnElement(_NULL_,INPUT,2); WaitPages(1); FindFormByName(search,0); SetInputValue(name,0,@ARTICLE); ClickOnElement(search,INPUT,0); WaitPages(1); |

the browser. The web wrapper stores as “initialization” time the effort required to access the home page (where the query form resides); the “first result list” time stores the time required to query the web page and return the first page of results. The “following results time” stores the time needed to retrieve the rest of results (up to a configurable depth). Regarding Citeseer [2], the web wrapper arrives to the home page (“initialization” time) and queries the application, but for each result, it also accesses their detailed information page, which all sums up to the “first result list” and “following results” time where required.

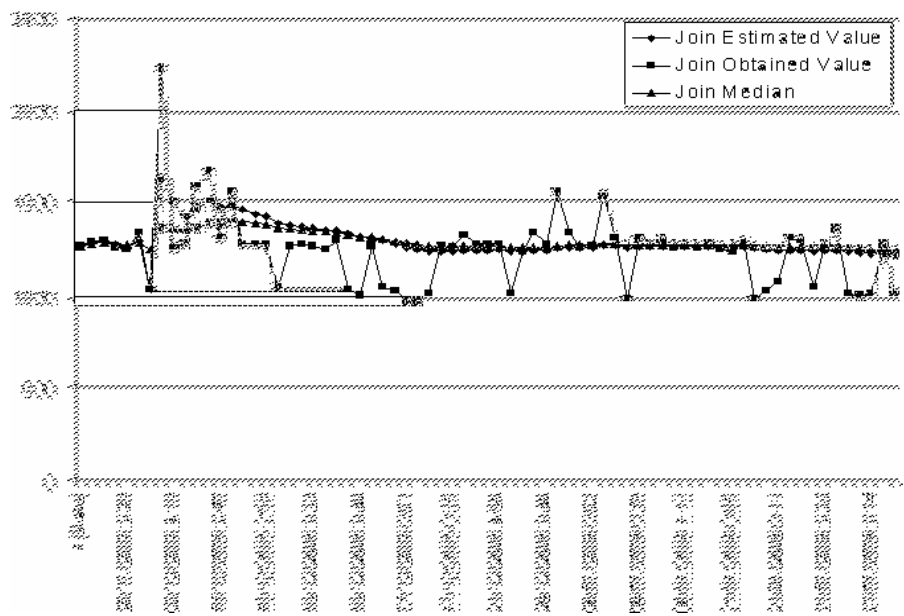


Fig. 2. Join between two sources

For the test depicted here, we process the list of research articles returned by a specific query on either Google Scholar or Citeseer, since, for this particular experiment, we consider that these sources have the same quality and, are therefore supplementary. This information will be matched against an intranet application with information about current papers published by the department, and their citations, that is, a join operation (with two possible operator strategies: hash join or nested-loop join); this source requires authentication and must be queried with a set of keywords describing the title of the original paper which citations must be returned. Therefore, the basic query to be performed is `SELECT * FROM RESEARCH JOIN ON CITATIONS WHERE SEARCH = 'value'`, and where RESEARCH is either Google Scholar or Citeseer. The use of external and internal web sources with regards to a department or company is a current need in applications using mediator architectures, such as Competitive Intelligence or Corporate Search.

**Fig. 2** describes the evolution of the total cost of the chosen query plan over a time period of three months (between November, 2005 and January, 2006), in a particular time section (from 15:00 to 00:00 GST -1). The figure shows the estimated cost of the chosen query plan (the diamond-dotted line, named as “Join Estimated Value”), the actual cost after the wrapper execution (the square-dotted line, “Join Obtained Value”), and the estimated cost of the query plan when base relation cost values are retrieved by using an arithmetic median (triangle-dotted line, “Join Median”). In this experiment Google Scholar was always the best option over Citeseer; this is because Citeseer required accessing every single detailed information page for each result in the list, while the output schema required for Google Scholar could be filled just by accessing the result list page. Besides, the optimal join operator strategy was the hash join in almost 100% of the times, except when the citations web source was artificially modified in order to return the following results at a much slower pace than citeseer; as explained in

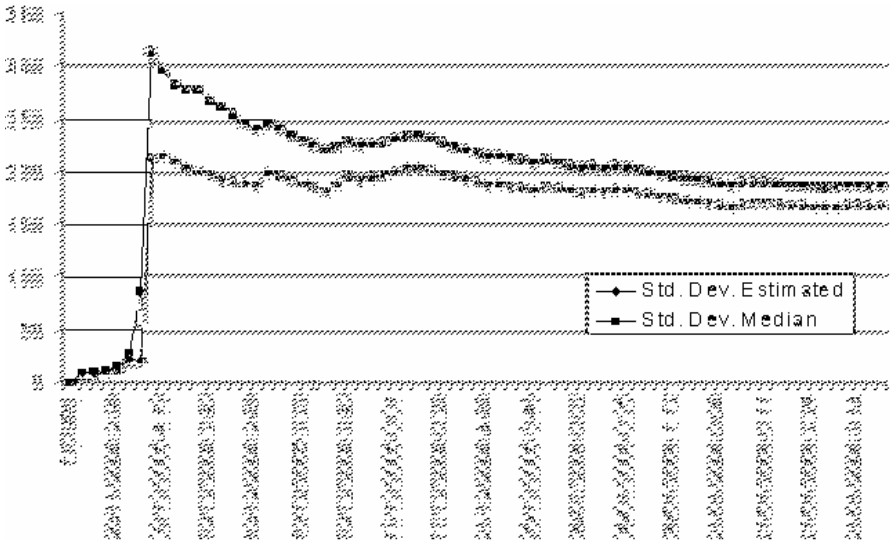


Fig. 3. Standard Deviation of Proposed Technique and Median

section 4, the nested-loop join behaves better if the time required to access the first source, and the “first results” time of the second source are much lower than the “following results” time of the second source, that is, when :

$$\begin{aligned} \text{totalTime}(\text{source}_1) &< \text{totalTime}(\text{source}_2) , \text{ and} \\ \text{totalTime}(\text{source}_2) &> \text{totalTime}(\text{source}_1) + \\ &\text{firstResultsTime}(\text{source}_2) \cdot \text{ceiling}(N \cdot \text{tupleNumber}(\text{source}_1)) . \end{aligned} \quad (8)$$

**Error! Reference source not found.** shows how the cost obtained by the propagation plus updating techniques behaves better than the median estimating the cost of the operation (triangle-dotted line depicts the standard deviation of the proposed technique, while, the square-dotted line shows the standard deviation of the median). It can be observed that the model reacts very rapidly to sudden changes of the query sample, and the values obtained are always better than those from the arithmetic mean. Thus, our method is shown to have a better efficiency on the number of query samples, and a better behaviour.

## 7 Conclusions

Optimization of mediators is critical as a way of improving performance in real-time access to remote sources. In this paper we propose the use of a cost model which stores a set of parameters for each query, and a novel statistical technique which updates each parameter everytime a new value arrives to the mediator as measured by the wrapper, with a constant efficiency  $\theta(1)$  on the number of stored queries. The set of parameters chosen for the model has been defined explicitly for web sources, even though they could be easily used in other scenarios; these parameters have been tested in real-world environments, with different kinds of sources, and have shown their usefulness and completeness.

While deciding what parameters to use is a very important issue in query optimization, since a bad choice can lead to a wrong choice of query plans, updating the value of each parameter every time a new value arrives is absolutely critical. Using an outdated model to propagate the base view costs up to each candidate query plan can be very harmful due to the uncontrolled context in which web sources operate, with no communication reliability between the mediator and the sources. Thus, an adequate parameter updating technique, theoretically valid and with high performance, is a must in real environments in which tenths or hundreds of queries are executed per unit of time. The parameter updating technique developed in this paper is demonstrated to be very efficient, since each new model is independent from the number of values stored in the repository.

We have performed a thorough set of experiments on our implementation, using real-world web sources to test the adequacy of the model and its performance. The experimental results are in accordance with the theory developed previously, showing an important gain in how the mediator updates the cost repository and chooses the most promising query plan from the tentative ones.

The features shown in this paper have been implemented on Denodo’s current Enterprise Information Integration product, Virtual DataPort [10].

## References

1. Adali, S., Candan, K., Papakonstantinou, Y., Subrahmanian, V.S. Query Caching and Optimization in distributed mediator systems. ACM SIGMOD Conference on Management of Data. 1996
2. Citeseer Scientific Literature Digital Library. <http://citeseer.ist.psu.edu>
3. Du, W., Krishnamurthy, R., Shan, M. Query Optimization in a Heterogeneous DMBS. 18th International Conference on Very Large Databases. 1992
4. Google Scholar Search Engine. <http://scholar.google.com>
5. Hidalgo, J., Pan, A., Losada, J., Álvarez, M. Adding Physical Optimization to Cost Models in Information Mediators. 2005 IEEE Conference on e-Business Engineering. 2005
6. Hidalgo, J., Pan, A., Losada, J., Álvarez, M., Viña, A. Building the Architecture of a Statistics-based Query Optimization Solution for Heterogeneous Mediators. 6th International Conference on Information Integration and Web-based Applications & Services. 2004
7. Ives, Z., Florescu, D., Friedman, M., Levy, A., Weld, D. S. An Adaptive Query Execution System for Data Integration. 24<sup>th</sup> International Conference on Very Large Databases. 1999
8. Naacke, H., Gardarin, G., Tomasic, A. Leveraging mediator cost models with heterogeneous data sources. ICDE. 1998
9. Ozcan, R., Haas, L. Cost Models DO Matter: Providing Cost Information for Diverse Data Sources in a Federated System. The VLDB Conference. 1999
10. Pan, A., Raposo, J., Álvarez, M., Montoto, P., Orjales, V., Hidalgo, J., Ardao, L., Molano, A., Viña, A. The DENODO Data Integration Platform. 28th International Conference on Very Large Databases. 2002
11. Papakonstantinou, Y., Gupta, A., Haas, L. Capabilities-Based Query Rewriting in Mediator Systems. Fourth International Conference on Parallel and Distributed Information Systems. 1996
12. Rahal, A., Zhu, Q., Larson, P-A. Evolutionary Techniques for Updating Query Cost Models in a Dynamic Multidatabase Environment. The VLDB Journal – The International Journal on Very Large Data Bases. Vol. 13, No. 2, 2004.
13. Raposo, J., Pan, A., Hidalgo, J., Viña, A. The WARGO System: Semi-automatic Wrapper Generation in presence of Complex Data Access Modes. 2nd International Workshop on Web Based Collaboration. 2002
14. Wiederhold, G. Mediators in the Architecture of Future Information Systems. IEEE Computer, March 1992, pp. 38-49
15. XIWT. Cross-Industry Working Team. Internet Service Performance: Data Analysis and Visualization. July 2000. <http://www.xiwt.org/documents/IPERF-paper2.pdf>
16. Zadorozhny, V., Raschid, L., Vidal, M.E. Efficient Evaluation of Queries in a Mediator for WebSources. ACM SIGMOD International Conference on Management of Data. 2002

# TupleRank: Ranking Discovered Content in Virtual Databases

Jacob Berlin and Amihai Motro

Information and Software Engineering Department  
George Mason University, Fairfax, VA 22030, USA  
{jberlin, ami}@gmu.edu

**Abstract.** Recently, the problem of data integration has been newly addressed by methods based on machine learning and discovery. Such methods are intended to automate, at least in part, the laborious process of information integration, by which existing data sources are incorporated in a virtual database. Essentially, these methods scan new data sources, attempting to discover possible mappings to the virtual database. Like all discovery processes, this process is intrinsically probabilistic; that is, each discovery is associated with a specific value that denotes assurance of its appropriateness. Consequently, the rows in a discovered virtual table have mixed assurance levels, with some rows being more credible than others. We argue that rows in discovered virtual databases should be *ranked*, and we describe a ranking method, called *TupleRank*, for calculating such a ranking order. Roughly speaking, TupleRank calibrates the probabilities calculated during a discovery process with historical information about the performance of the system. The work is done in the framework of the Autoplex system for discovering content for virtual databases, and initial experimentation is reported and discussed.

## 1 Introduction

Recently, the problem of data integration has been newly addressed by methods based on *machine learning* and *discovery* [2, 3, 7, 5, 9]. Such methods are intended to automate, at least in part, the laborious process of information integration, by which existing data sources are incorporated in a virtual database. Essentially, these methods scan new data sources, attempting to discover possible mappings to the virtual database.

Like all discovery processes, this process is intrinsically probabilistic; that is, each discovery is associated with a specific value that denotes assurance of its appropriateness. Consequently, the rows in a discovered virtual table have mixed assurance levels, with some rows being more credible than others. This suggests that the content of discovered virtual databases should be *ranked*.

Such ranking is useful in several ways. First, users retrieving data from discovered virtual databases should be made aware of their associated “quality,” as such information is essential for any decision process that might be based on the data. Moreover, users could now retrieve from discovered virtual databases on the basis of quality; for example, only information that exceeds a certain level of quality is retrieved.

Finally, ranking of discovered content permits fully-automated data integration. Typical data integration systems may be classified at best as semi-automatic [4, 7, 9, 10], requiring domain experts to refine each discovered mapping into a “correct” mapping. By developing discovery methods that annotate their discoveries with accurate predictions of their “correctness,” and then using these values for ranking the discoveries, we can offer an automated alternative. In other words, whereas other systems guarantee “perfect” mappings through the use of experts, our approach admits mappings of varying quality, relying instead on a grading method.

In this respect, our approach resembles the approach adopted by most information retrieval systems (and virtually all Internet search engines), where large magnitude searches are now performed automatically, and sophisticated answer-ranking methods substitute for the authoritative answers previously provided by human search experts.

The ranking method used in Autoplex is based on two measurements. The first measurement, called *assurance*, is calculated during the discovery process. Essentially, it is a probabilistic measurement which is calculated for each discovery and indicates how well this discovery satisfies expectations (these expectations are based on various forms of knowledge, both acquired and declared). Assurance is the subject of Section 3. By itself, though, assurance may be misleading. To illustrate, imagine an information retrieval expert who examines the abstract of a document and issues a value between 0 and 1 to indicate the document’s relevance to a particular search request. The predictions of the expert may be systemically biased; e.g., too high, too low, or involving a more complex bias. Comparing this expert’s predictions with the actual relevance of the documents (established, for example, by examining the actual documents), provides a means for *calibrating* this expert’s predictions. The second measurement, called *credibility*, is an indication of the actual performance of the Autoplex system. The calculation of this measurement, and the way in which it is used to calibrate the individual assurance measurements are the subjects of Section 4. The outcome of this process is an individual credibility estimate for each discovery.

These credibility estimates become part of the virtual database system, and are applied to rank every virtual table that is materialized by the system. The ranking method, called TupleRank, is a relatively simple derivative of the estimates and is described in Section 5. The TupleRank methodology has been validated in initial experiments. The experiments and the validation methodology are the subject of Section 6. Section 7 summarizes the findings and outlines ideas for further research. We begin with a brief discussion of virtual databases and the Autoplex discovery system for virtual databases.

## 2 Background

Our work is conducted within the framework of the Multiplex virtual database system [11] and the Autoplex content discovery system [2], and in this section we provide brief descriptions of both. We note, however, that much of our work is of general applicability to most such database integration systems.

## 2.1 Multiplex

The multidatabase system Multiplex [11] is an example of a virtual database system. The basic architecture of Multiplex is fairly simple (Figure 1). Whereas a traditional database consists of *schema* and *data* that conforms to the schema, a Multiplex database consists of *schema* and *mapping* that describes data that is stored elsewhere. More specifically, the schema is described in the relational model and the mapping is a list of *contributions*. Each contribution is a pair of expressions: the first expression is a view of the schema (expressed in SQL); the second is a query to one of the member databases (expressed in the language of that system). The answer to the second expression is assumed to be a materialization of the view described in the first expression. The complexity of these expressions can vary greatly: they could range from a complex calculation, to a statement that simply denotes the equivalence of two fields. The requirement that the second expression is written in the language of the source supports member heterogeneity. Alternatively, heterogeneity can be supported by using *wrappers*, which are software modules that provide a uniform interface to the individual sources [8]; in which case these expressions would be written in a single system-wide language.

The main challenge is to decompose a user query into queries to the individual databases, and then combine the individual answers in an answer to the original query, thus providing complete *transparency*. Additional challenges are to provide *partial* answers in the absence of some of sources, and to resolve *inconsistencies* when sources provide overlapping information.

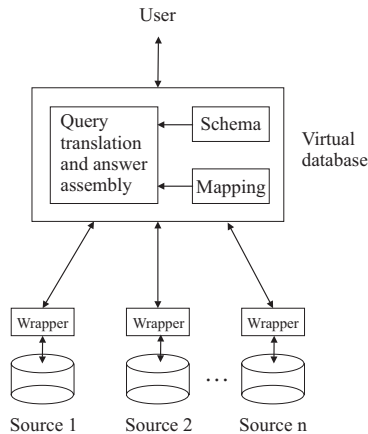


Fig. 1. Multiplex architecture

## 2.2 Autoplex

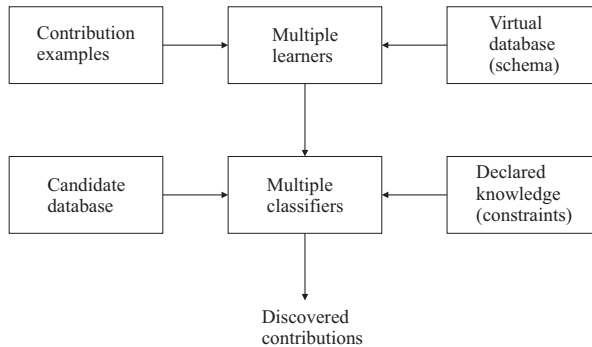
A significant limitation of virtually all such multidatabase systems is the difficulty of mapping new sources into the virtual database. Recognizing that



mapping *complete* member schemas is not a viable approach, the Multiplex methodology allows for *ad hoc* mapping of *selected* parts of the sources. Still, in an environment in which the set of member sources is very large and constantly changing, this methodology may not scale up very well.

To address this limitation we have been developing a system called Autoplex [2], for *discovering* member schemas and incorporating them into the virtual schema with only limited human effort. Based primarily on supervised learning, the system acquires knowledge from examples that have already been integrated into the virtual database. With this acquired knowledge, the system can extract “content contributions” from new information sources.

A high level overview of the Autoplex architecture is shown in Figure 2. This architecture includes two main components: a suite of *learners* and a suite *classifiers*. The learners acquire knowledge from examples; the classifiers use this acquired knowledge, as well as various forms of pre-declared knowledge, to infer a contribution from a new source.



**Fig. 2.** Autoplex architecture

Each learner is given a virtual database  $D$  consisting of tables  $R_1, \dots, R_n$ , and a set of contribution examples. Each such example consists of a local scheme  $S$ , an instance of this scheme  $s$ , and a contribution pair: a selection-projection expression on  $S$  and a selection-projection expression on  $R$ . The extension of the expression on the instance  $s$  generates rows for the expression on the virtual table  $R$ . The learner uses this information to acquire knowledge on features of the examples. This knowledge is stored on secondary storage in efficient data structures for future use.

A candidate scheme  $T$  and instance  $t$  are provided by a new member database as inputs to the corresponding classifier. This classifier uses the acquired knowledge in conjunction with pre-declared knowledge (e.g., constraints) to infer a selection-projection view that defines a contribution of  $T$  to a table  $R$  in the virtual database. The classifier also outputs assurance metadata that can be used for ranking the discovered content.

The use of multiple learners and classifiers was first applied in the area of database integration in [6, 7], and we use a comparable approach in Autoplex.

Currently, we use learners and classifiers for both domain values and patterns. Furthermore, we distinguish between learners and classifiers for columns and learners and classifiers for rows. This distinction is necessary to support our search for contributions, to be discussed in the next section.

### 3 Discovery and Assurance

In this section we review the Autoplex content discovery process. In particular, we explain the derivation of the assurance measurement for each discovery.

Autoplex searches for *transformations* of both a candidate table and a virtual table such that the former is a materialization of the latter. For reasons of efficiency, transformations are limited to *projections* and *selections* only. A projection removes columns from a table and possibly reorders the remaining columns; a selection removes rows from the candidate table based on a row predicate.

Furthermore, Autoplex considers projective and selective transformations in two *separate* phases. First, it finds the best projective transformation over a threshold (if any), then, it attempts to improve it by the best possible selective transformation (if any).

#### 3.1 Projection

Finding projective transformations of both the candidate and virtual tables so that the resultant tables are semantically equivalent amounts to finding a *matching* (a one-to-one mapping) between a subset of the columns of the candidate table and a subset of the columns of the virtual table. Let  $\pi$  denote a possible column matching. Autoplex uses a rather complex formula to score the goodness of this matching. A *hill-climbing* search is then used to optimize this formula; i.e., to find the column matching with the highest score. We briefly discuss the Autoplex scoring of column matchings.

The Autoplex score of each column matching is based on the output of multiple classifiers. These classifiers fall into two categories: classifiers that apply *acquired knowledge* and classifiers that apply *declared knowledge*. The former type of classifiers are based on traditional machine learning principles. The primary Autoplex learner is a domain learner. This learner acquires from training examples probabilistic information on the affinities between domain values and virtual columns. Given a new column of values, the corresponding classifier uses this information to score the association between this column and each of the virtual columns. Other learners may be based on column names, string patterns, and so on.

The learners and classifiers consider only individual columns, ignoring any possible interactions among columns. At times, however, the examples may feature such interactions; for instance, the examples may indicate that two virtual columns  $A$  and  $B$  have a relationship of monotonicity. Learning and classifying multi-column relationships is computationally prohibitive, and for such features Autoplex relies on declared knowledge. Declared knowledge is simply

a pronouncement of relationships among virtual columns (e.g., primary keys, monotonicity, association rules). For every item of declared knowledge, a corresponding classifier scores the level of satisfaction of each column matching. Such classifiers will also be referred to as constraint checkers.

The classifiers in each group are given weights, designating their relative importance or credibility. Because the acquired-knowledge classifiers assign scores to individual column matches, the overall knowledge about each matching may be represented in a bipartite graph. The columns of the candidate and virtual tables are nodes in two separate partitions, and a matching is a set of edges connecting nodes in the two partitions (no two edges share a node). Each edge is labeled with the weighted score of the available classifiers, and the overall score is the product of these labels. For a matching  $\pi$  this score will be denoted  $P(\pi)$ . For the declared-knowledge classifiers, the overall score is obtained by weighting the scores of the individual constraint checkers. This score will be denoted  $S(\pi)$ . Finally, these two scores are weighted with a constant  $\alpha$ , resulting in this formula

$$ScoreColumn(\pi) = \alpha \cdot S(\pi) + (1 - \alpha) \cdot P(\pi) \quad (1)$$

The optimization process starts with a random, complete matching of the candidate columns to the virtual columns. It then iteratively improves the quality of this matching by swapping and removing edges in the matching.<sup>1</sup> If the best score exceeds a threshold, then Autoplex proceeds to the next phase.

### 3.2 Selection

The purpose of this phase is to extract an optimal set of rows from the candidate table, and then describe it with an abstract predicate. Let  $\sigma$  denote a subset of rows. Here, too, Autoplex creates a scoring formula for  $\sigma$  that combines verdicts of acquired-knowledge classifiers with declared-knowledge classifiers (constraint checkers).

Recall that the primary learner and classifier in the projection phase were essentially aimed at domains. That is, the learner associated each virtual column with a domain of values, and the classifier matched a candidate column to the most similar domain. Even though the examples show that specific combinations of values (tuples) are more likely to occur than others, this learner has ignored such information. In other words, the corresponding classifier will consider all tuples in the Cartesian product of the domains to be equally likely. The primary learner and classifier in the selection phase address this issue. The learner acquires probabilistic knowledge on the example rows that are selected and those that are discarded, and the classifier assigns each candidate row a probability of being a contribution. In the presence of additional acquired-knowledge classifiers, their verdicts on each row are weighted together to provide a single score.

---

<sup>1</sup> Note that in the absence of declared-knowledge classifiers,  $ScoreColumn(\pi)$  is simply  $P(\pi)$  and the problem is reduced to finding a maximal weighted matching in a bipartite graph, for which an efficient  $O(n^3)$  algorithm is available.

The score of a subset  $\sigma$  of the candidate rows, denoted  $P(\sigma)$ , is then a combination of these individual row scores. Finding an appropriate method of combination presents a small challenge. The obvious method of *totaling* the scores of the participating rows is flawed because it encourages the subsequent optimization process to select the entire table. Similarly, *averaging* the scores of the participating rows is flawed as well because it encourages the optimization process to select only the highest scoring row. A good solution appears to be a *product* of scores, in which each row selected contributes its score  $w$ , and each row discarded contributes its complement score  $1 - w$ .

This subset of rows  $\sigma$  is also assigned a score by each available declared-knowledge classifier (constraint checker), reflecting its level of compliance, and these scores are weighted together to provide a single satisfaction score  $S(\sigma)$ . The scores  $P(\sigma)$  and  $S(\sigma)$  are then weighted with a constant  $\beta$ . Altogether, the formula

$$ScoreRow(\sigma) = \beta \cdot S(\sigma) + (1 - \beta) \cdot P(\sigma) \quad (2)$$

defines the overall score Autoplex assigns to the prospect that  $\sigma$  is the set of rows that contribute to the virtual table.

As in the projection phase, this scoring function is optimized in a hill-climbing search which begins with a random subset of the rows and improves it iteratively. This optimization process results in set of rows that contribute to the virtual table and a set of discarded rows. The rows are labeled accordingly and a standard classification tree algorithm<sup>2</sup> is applied to find a selection predicate that defines the contributing set of rows. This extension-independent definition will be used to extract the appropriate set of rows should the source be updated.

The constants  $\alpha$  and  $\beta$  in the two scoring functions (Equations 1 and 2) denote our willingness to discount the recommendations of the acquired-knowledge classifiers for better constraint satisfaction (and vice versa). This technique of combining general constraints with mapping decisions is due to [7]; however, we extend it to search for both projective and selective transformations.

### 3.3 Assurance

Finally, the two scores generated in the two phases of the discovery are combined in a single measurement of assurance. Both scores are highly influenced by the cardinalities of the discovery (the number of columns and rows, respectively), with higher cardinalities resulting in lower scores. To compensate, these two scores are normalized by their cardinalities. Let  $n$  and  $m$  indicate the number of columns and rows, respectively, in the discovery. Assurance is defined as

$$Assurance(\pi, \sigma) = ScoreColumn(\pi)/n + ScoreRow(\sigma)/m \quad (3)$$

Because *ScoreColumn* and *ScoreRow* are generated from products of probabilities and such products tend to become very small, these probabilities are mapped to a logarithmic scale. Consequently, assurance values are always negative.

<sup>2</sup> Autoplex uses the algorithm J48 from the WEKA machine learning package [12].

## 4 Credibility and Calibration

The assurance measurement is an indication of how well each Autoplex discovery meets expectations. To correct for any systemic biases in this measurement, it is calibrated by the actual performance of the system, termed credibility. We discuss first how credibility is measured, and then how assurance and credibility are combined in a single estimate.

### 4.1 Measuring Credibility

The success of any discovery process must be judged on the basis of two criteria: (1) its *soundness* — the proportion of the cases discovered that are correct, and (2) its *completeness* — the proportion of the correct cases that are discovered. In Autoplex, a discovery is a mapping of a candidate table to a virtual table, and it is judged with respect to an expert mapping of these tables. Hence, the credibility of Autoplex is measured by *two* dual values.

Soundness and completeness are calculated for every candidate table at the level of the *cell*. Let  $t$  be a candidate table, let  $t_d$  be the subtable *discovered* by Autoplex and let  $t_e$  be the subtable *extracted* by an expert. Together, soundness and completeness measure the overlap of these two subtables. A cell discovered (i.e., in  $t_d$ ) and also extracted (in  $t_e$ ) contributes to soundness, whereas a cell extracted (in  $t_e$ ) and also discovered (in  $t_d$ ) contributes to completeness.<sup>3</sup> Specifically, the cells of  $t$  falls into four disjoint categories:

- $A$  = Cells that are both discovered and extracted (*true positives*).
- $B$  = Cells that are extracted but not discovered (*false negatives*).
- $C$  = Cells that are discovered but not extracted (*false positives*).
- $D$  = Cells that are neither discovered nor extracted (*true negatives*).

The soundness and completeness of a discovery is calculated from the cardinalities of the first three categories:

$$Soundness = \frac{|A|}{|A| + |C|} \quad (4)$$

$$Completeness = \frac{|A|}{|A| + |B|} \quad (5)$$

Soundness is the proportion of valid cases among the discovered cells, and thus measures the accuracy of the Autoplex discovery process. Completeness is the proportion of discovered cases among the valid cells, and thus measures the ability of Autoplex to discover cells. Both measures fall on the  $[0, 1]$  interval, and higher values are better.

Soundness and completeness may be viewed also as probabilities. Soundness is the probability that a discovery is a valid contribution, whereas completeness is the probability that a valid contribution is discovered.

---

<sup>3</sup> We use the primary key of the table to uniquely identify rows and thus cells.

Soundness and completeness correspond to the *precision* and *recall* measures, which are used widely in information retrieval [1]. In some applications, a single performance measure may be preferred, and various ways have been suggested to combine these measures into a single measure (e.g., using their harmonic mean). Here, however, we shall measure the performance of discoveries using both.

## 4.2 Calibration

The credibility measurements of soundness and completeness are used to *calibrate* the assurance measurement derived in the discovery process. This calibration process uses a set of “training” tables and a numeric discretizing technique from data mining called *equal frequency binning* [12]. The intuition behind this approach is that, while the calculated assurance scores may be inaccurate, higher assurance scores result in higher credibility (i.e., assurance scores are positively correlated to soundness and completeness).

Roughly speaking, the calibration process may be summarized as follows: If the track record of Autoplex shows that when it discovered content with assurance  $x$ , its credibility was  $y$ , then future discoveries made with assurance  $x$  will be estimated to have credibility  $y$  as well. A more formal description of the calibration process is given in this 7-step procedure:

1. Autoplex is tasked to discover contributions in the training tables.
2. Experts are used to extract optimal contributions from the same tables.
3. The soundness and completeness of each discovery is calculated.
4. The discoveries are sorted according to their assurance scores.
5. The sorted list of discoveries is divided into  $b$  “bins” of equal size.
6. The soundness and completeness of each bin are calculated as the *aggregate* soundness and completeness of all the discoveries in that bin.<sup>4</sup>
7. Finally, boundaries between adjacent bins are calculated by averaging the scores of discoveries that separate the bins.

Since not all training tables are of equal size, equal sized subsets of these tables are used during the discovery phase; this ensures that the bins are all of the same size.

Given a new candidate table, Autoplex discovers the optimal contribution as explained in Section 3. The assurance score and the boundaries (calculated in Step 7) are used to locate the appropriate bin, and the soundness and completeness values of the bin (calculated in Step 6) are associated with the discovery as estimates of its credibility.

Results from actual experimentation in Autoplex are summarized in Table 1. The experiment involved 170 training examples. For each example, assurance and credibility (soundness and completeness) were measured. These measurements showed strong positive correlation between assurance and soundness (0.83) and

---

<sup>4</sup> *Aggregate* soundness and completeness are obtained from the *unions* of the true positives, false positives and true negatives of the different discoveries, and are usually more informative than *average* soundness and completeness.

between assurance and completeness (0.86), thus validating our working assumption. The 170 examples were assigned to 10 equal-size bins. For each bin, Table 1 shows the boundary assurance value and the aggregate soundness and completeness of the 17 discoveries in the bin. Now consider a future discovery made with assurance -0.8. This discovery will be assigned to bin 4 and will be estimated to have soundness 0.6486 and completeness 0.8205.

**Table 1.** Example of calibration data

| Bin | Assurance<br>(lower boundary) | Soundness<br>(aggregate) | Completeness<br>(aggregate) |
|-----|-------------------------------|--------------------------|-----------------------------|
| 1   | -0.1232                       | 0.8715                   | 0.9623                      |
| 2   | -0.5977                       | 0.7843                   | 0.9388                      |
| 3   | -0.7704                       | 0.7279                   | 0.9778                      |
| 4   | -0.9638                       | 0.6486                   | 0.8205                      |
| 5   | -1.3222                       | 0.5410                   | 0.8081                      |
| 6   | -1.7155                       | 0.3399                   | 0.7128                      |
| 7   | -1.9522                       | 0.0785                   | 0.5375                      |
| 8   | -2.4041                       | 0.0763                   | 0.3167                      |
| 9   | -2.7884                       | 0.0241                   | 0.2281                      |
| 10  | (none)                        | 0.0109                   | 0.0784                      |

Recall that each contribution is an entry in the mapping of the virtual database (Figure 1). Autoplex discovered contributions are added to the same mapping, but with additional annotations on their estimated credibility.

## 5 Ranking

The credibility estimates of the discoveries can be used in several ways, and in this section we sketch three possibilities.

A relatively simple deployment of these estimates is to specify in retrieval requests *thresholds of credibility* and require that the evaluation of these requests be based only on discoveries whose credibility exceeds these thresholds. In this way, users can guarantee minimal performance for their queries. If necessary (for example, if the answers that are retrieved are deemed to be too small or too large), these thresholds may be relaxed or strengthened.

A second way in which these estimates can be used is to calculate the *credibility estimates of each answer* that is issued by the system. The main concern here is how to combine the credibility estimates of the different contributions that are used in generating an answer. A straightforward approach is to weigh the estimates of the contributions according to their level of participation in the answer.

A third, and possibly most challenging, way in which the credibility estimates of contributions can be used is to calculate the *credibility of individual rows* of each answer, and then *rank the rows* accordingly. This provides users with information on the quality of each row, and allows them to adopt rows of

higher quality first. We refer to the Autoplex method of ranking answer rows as *TupleRank*.

TupleRank requires an estimate of the credibility of each individual cell of an answer. Of the two measures used to estimate the credibility of contributions, only soundness can be propagated to individual cells. This is because the proportion of discovery cells that are correct can be translated to a probability of each cell being correct; but the proportion of the correct cells in a candidate table that are included in a discovery cannot be translated into a property of the individual cells.

Each cell of an answer is derived from one or more contributions and inherits their credibility. If a cell is obtained from multiple contributions then its soundness is calculated as the average of the soundness of these contributions. The soundness of each row is then calculated as the average soundness of its cells. This final value is used for ranking the answer.

The three methods discussed in this section may be combined. Given a query, only a qualifying set of contributions is used to evaluate its answer; the soundness of each individual row is calculated and the answer is ranked; the sorted answer is accompanied by its overall credibility estimates.

## 6 Experimentation and Validation

The purpose of the experiment we describe in this section is to validate our calibration methods; that is, to show that calibration improves predictions. Without the use of calibration, the best prediction that can be made regarding the credibility of a discovery is the aggregate performance of Autoplex on a set of test data. Furthermore, this prediction is the same for every discovery. We call this the *uncalibrated prediction*. With calibration, predictions can be made for individual discoveries. We call this the *calibrated prediction*.

To validate the effectiveness of calibration, we ran Autoplex on test data that an expert has mapped into the virtual database. We compared the uncalibrated and calibrated predictions of soundness and completeness relative to the expert mappings, with the expectation that calibrated predictions will be more accurate.

Higher accuracy corresponds to lower error, so we measured the error of uncalibrated and calibrated predictions. Error was measured using Root Mean Squared Error (RMS), which is commonly used to evaluate the effectiveness of numeric prediction. RMS is defined as

$$\sqrt{\frac{\sum_{i=1}^n (p_i - a_i)^2}{n}} \quad (6)$$

where  $p_i$  and  $a_i$  are the predicted and actual soundness (or completeness) for the  $i$ th discovery of  $n$  discoveries. The RMS error for soundness and completeness predictions falls in the  $[0, 1]$  interval, and lower values are better.

Two separate experiments were conducted. One experiment used a virtual database on computer retail (originally used in [2]):<sup>5</sup>

---

<sup>5</sup> Primary keys are indicated in *italics*.



1. Desktops = (*Retailer*, *Manufacturer*, *Model*, Cost, Availability)
2. Monitors = (*Retailer*, *Manufacturer*, *Model*, Cost, Availability)
3. Printers = (*Retailer*, *Manufacturer*, *Model*, Cost, Availability)

The other experiment used a virtual database on real estate (reported in [7]):

1. Property = (*PropertyId*, Address, Description, Price, Bedrooms, Bathrooms)
2. Agents = (*AgentName*, AgentPhone, AgentEmail)

To experiment with this data, we used a procedure from data mining called *stratified threefold cross-validation* [12], which we briefly describe. Each of the sources was manually mapped into our virtual database by using a mapping table as discussed in Section 2.1. We partitioned the mappings in our mapping table into three folds of approximately equal content. Using two folds for learning and calibration and one fold for testing (i.e., discovery), we repeated the experiment for the three possible combinations of folds. To measure the soundness and completeness of the discoveries, the information in the mapping table was assumed to be the correct mapping of these sources.

For each discovery we measured the error of the calibrated and uncalibrated predictions relative to the correct mappings. Table 2 summarizes the results of the experiments. Both experiments show that the calibrated credibility measures are always more accurate than the uncalibrated ones. Evidently, in three of the four cases the improvement is considerable.

**Table 2.** RMS error of predicted vs. actual credibility

| Credibility               | Computer Retail | Real Estate |
|---------------------------|-----------------|-------------|
| Calibrated Soundness      | 0.22            | 0.31        |
| Uncalibrated Soundness    | 0.36            | 0.47        |
| Calibrated Completeness   | 0.24            | 0.29        |
| Uncalibrated Completeness | 0.34            | 0.30        |

## 7 Conclusion

Recent methods for automatically discovering content for virtual databases result in databases with information of mixed credibility. In this paper we argued that it is important to estimate the credibility of discovered information, so that these estimates could be used to calculate credibility estimates for all information issued from the virtual database. We defined credibility as a pair of measures, soundness and completeness, and we showed how to calculate reliable credibility estimates for each discovery, based on values calculated during the discovery process, and the actual performance of the system on test data. We also sketched various methods in which credibility estimates can be applied.

With respect to experimentation, we have shown so far that the Multiplex virtual database approach is viable [11], that the Autoplex automatic discovery

approach is attractive [2], and (in this paper) that the credibility of discoveries can be estimated reliably. Our focus now is on integrating these results in a single system; that is, incorporate the Autoplex methodology into Multiplex, and modify the Multiplex user interface to provide the type of capabilities described in Section 5.

Undoubtedly, the estimation of the credibility of discoveries and the ranking of tables formed from different discoveries bring to mind information retrieval systems, such as Internet search engines. Extending the Multiplex/Autoplex methodology to discover content in Web pages with semi-structured data (e.g., represented in XML) is a subject currently under investigation.

## References

1. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley/ACM Press, 1999.
2. J. Berlin and A. Motro. Autoplex: Automated discovery of content for virtual databases. In *Proc. COOPIS 01, Ninth Int. Conf. on Cooperative Information Systems*, pages 108–122, 2001.
3. J. Berlin and A. Motro. Database schema matching using machine learning with feature selection. In *Proc. CAiSE 02, Fourteenth Int. Conf. on Advanced Information Systems Engineering*, pages 452–466, 2002.
4. S. Castano and V. De Antonellis. A schema analysis and reconciliation tool environment for heterogeneous databases. In *Proc. IDEAS 99, Int. Database Engineering and Applications Symposium*, pages 53–62, 1999.
5. R. Dhamankar, Y. Lee, A. Doan, A. Y. Halevy, and P. Domingos. iMAP: Discovering complex semantic matches between database schemas. In *Proc. SIGMOD 04, Int. Conf. on Management of Data*, pages 383–394, 2004.
6. A. Doan, P. Domingos, and A. Y. Halevy. Learning source description for data integration. In *Proc. WebDB*, pages 81–86, 2000.
7. A. Doan, P. Domingos, and A. Y. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *Proc. SIGMOD 2001, Int. Conf. on Management of Data*, pages 509–520, 2001.
8. H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. D. Ullman, V. Vassalos, and J. Widom. The TSIMMIS approach to mediation: Data models and languages. *J. Intelligent Information Systems*, 8(2):117–132, 1997.
9. W.-S. Li and C. Clifton. SEMINT: A tool for identifying attribute correspondences in heterogeneous databases using neural networks. *Data & Knowledge Engineering*, 33(1):49–84, 2000.
10. J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with Cupid. In *Proc. VLDB 2001, 27th Int. Conf. on Very Large Databases*, pages 49–58, 2001.
11. A. Motro. Multiplex: A formal model for multidatabases and its implementation. In *Proc. NGITS 1999, Fourth Int. Workshop on Next Generation Information Technologies and Systems*, pages 138–158, 1999.
12. I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 2000.

# Analysis of Queries Reaching SHIL on the Web - An Information System Providing Citizen Information

Gilad Ravid<sup>1</sup>, Judit Bar-Ilan<sup>2</sup>, Sheizaf Rafaeli<sup>3</sup>, and Shifra Baruchson-Arbib<sup>2</sup>

<sup>1</sup> Department of Industrial Engineering and Management  
Ben-Gurion University of the Negev, Beer Sheva, Israel  
gilad@ravid.org

<sup>2</sup> Department of Information Science  
Bar-Ilan University, Ramat-Gan, Israel  
barilaj@mail.biu.ac.il, baruchsl@mail.biu.ac.il

<sup>3</sup> Center for the Study of the Information Society  
University of Haifa, Haifa, Israel  
sheizaf@rafaeli.net

**Abstract.** SHIL on the Webp is the website of the Israeli Citizens' Advice Bureau. This is a high traffic site, where the sources of most of the external hits are result pages of general search engines. In this paper we analyzed these external hits with an emphasis on the specific queries submitted to the general search engines (e.g., Google and MSN). The findings provide an insight to the general interests of the Israeli web users related to citizens' information.

## 1 Introduction

The Israeli Citizens' Advice Bureau (SHIL – an acronym for its name in Hebrew: Sherut Yeutz Laezrach) is charged with providing the public with information about rights, social benefits, government and public services and civil obligations. It does so using sixty centers throughout Israel, phone hotlines and a very active website (<http://shil.info>). Information sources for the site include governmental publications and communiques, as well as popular press articles, suggestions and contributions from the public and constituent organizations. The site is updated almost daily. The website began operations in 1994; has undergone a number of major upgrades, and currently is a comprehensive site providing information on the above-mentioned topics. The current site contains interfaces in four languages, addresses both national and local audiences and topics, and has an active “backoffice” that facilitates constant updating of content. The site is accompanied by an extensive set of logs and tracking mechanisms. The topics are arranged in a hierarchical directory-like fashion. The site also has an internal search engine with advanced features that allow limiting the search to geographical areas and to subtopics. In addition the site has a very active forum, where the citizens' questions are answered by SHIL employees and volunteers. Figure 1 shows a part of the English language homepage.

Activity on the site approaches 10,000 pages read per day. This measure does not include the interactive use of forums. This is a level high enough to allow analysis



**Fig. 1.** The English language homepage of SHIL on the Web

of aggregate behavioral patterns. A closer look at the logfiles of the SHIL website reveals some surfing behavior insights of importance: Portals seem to matter less than is warranted by the attention accorded to them. About 84% of the requests for which the referrer field is external to SHIL [11] are from general search engines (e.g. Google and MSN). In the context of this paper requests from external referrers are called *external hits*, and external hits from search tools, where the referral URL contains a query (i.e., the user reached the site after submitting a query at the referral site) are called *external queries*. The *referer* (misspelled in the official HTTP specification) or the *referring page* is the URL of the previous page from which a link was followed [21].

Queries to general search engines are the main source of *external hits*. Users reach a specific page on the site linked from the search results page. In this paper, as a first step in analyzing the access logs, we analyze the *external queries* that lead the users to the SHIL site in order to provide an insight to the general interests of the Israeli web users related to citizens' information.

## 2 Related Studies

There are a number of studies analyzing search engine logs. One of the first major studies was based on a set of almost 1,000,000 queries presented to AltaVista during a 43 days period in August-September 1998 [15]. Findings of the study included data on the average number of terms (a single word or a phrase enclosed in quotation marks) per query (2.35). Jansen, Spink and Pedersen [8] compared the results of the Silverstein et al. study with a one day log (of 3,000,000 queries) on AltaVista in 2002. The percentage of single word queries decreased from 25.8% to 20.4%, and the most frequently appearing query changed from "sex" in 1998 to "google" in 2002 ("sex" was at fourth place in 2002). A subset of the queries of 2002 was categorized, and "people, places and things" was the most frequently occurring category (over 49%).

Excite's logs were extensively analyzed in a series of papers ([7, 14, 17, 19, 23] and a number of additional conference presentations). The results were based on four

sets: a set of about 50,000 queries from March 1997, a set of over a million queries from September 1997 and two other sets of similar sizes from December 1999 and May 2001 respectively. The results of the analyses included the number of terms per query (about 2.4 on the average, with an increase to 2.6 in the last set) and the percentage of users who used Boolean queries (between 5% and 10%) and data related to search sessions. Spink, Jansen, Wolfram, and Saracevic [17] compared the last two datasets, enabling them to identify a shift in the interests of the searchers from entertainment, recreation and sex to e-commerce related topics, like commerce, travel, employment and economy.

After Excite stopped being an independent search engine, the “Spink team” switched to analyzing logs from the search engine AlltheWeb [6,18]. The results are comparable, except for some differences in topics searched: less emphasis on e-commerce related issues and more on people and computers, but “sex” was still the second most popular search term). AlltheWeb users generated slightly more queries per sessions than Excite users. In the second AlltheWeb study (with data from 2002) there was an increase of single word queries from 25% to 33%.

Ozmutlu, Spink and Ozmutlu [12] carried out a time-of-day analysis of the search logs of Excite and AlltheWeb based on the local time at the server. For Excite the busiest hour of the day in terms of query arrival was between 9:00 and 10:00 AM, while for AlltheWeb the largest number of queries per hour arrived between 8:00 and 9:00 AM. Hourly traffic of the AOL search engine powered by Google was analyzed by Beitzel et al. [1], according to their findings the busiest hour of the day was between 9:00 and 10:00 PM.

Additional large scale Web search engine log analyses were carried out for a Korean search engine [13], where specific techniques were developed to handle language specific problems and also for Vivisimo [24]. Spink and Jansen discuss their search log analysis studies in their book [16] and compare the results of nine large search log analysis studies in a recent paper [7].

Next we review studies analyzing search logs of individual sites. One of the first single Web site search studies was carried out by Croft, Cook and Wilder [4]. They examined the usage of the THOMAS Web site, intended to provide government information to the general public on the Web. Jones, Cunningham and McNam [10] analyzed more than 32,000 queries that were submitted to the New Zealand Digital Library over a period of more than one year in 1996-7. Cacheda and Vina [2] analyzed the search logs of the Spanish Web directory BIWE based on a 16-day log from 2000. Wang, Berry and Yang [22] carried out a four-year longitudinal analysis of queries submitted to a university Web site. Chau, Fang and Sheng [3] analyzed the queries submitted to the Utah state government website during a period of 168 days in 2003. The content provided on the Utah state government website resembles the information available from SHIL on the Web.

Finally we mention two studies that analyzed the *referrer* field of web site logs (the page visited just before hitting a page on the site). Thelwall [20] analyzed the log of the site of the *Wolverhampton University Computer Based Assessment Project* for a period of ten months in 2000 and found that nearly 80% of the external hits were requests from search engines, most of them from Yahoo. Davis [5] studied the distributions of a set of referrals to the American Chemical Society’s site. The aim of the study was to understand how scientists locate published articles. In this case only 10% of the referrals were from search engines.

### 3 Research Questions, Data Collection and Methods

In this paper we study the *external queries*. Our aim is to understand:

- What citizen's information is of interest to users who arrive to SHIL on the Web from search engines?
- What are the characteristics of these queries? Are there specific problems because the site is multi-lingual?
- How do these queries compare to those submitted to general search engines and to queries submitted to local search engines, especially to the queries submitted to the Utah state government website [3]?

The dataset contained all the external hits to the SHIL website for an eight months period between March 1, 2005 and October 30, 2005. The original dataset comprised 306,383 records. Each record contained the IP address of the requester, the exact time and date of the request, the referring site, the referring query for *external queries* and the requested page on the SHIL site. The pages on SHIL website are organized into categories and articles within the categories, where the top level page in each category has no article id, and contains a linked list of all the articles in the specific category.

All requests with status codes other than 200 (successful) and 304 (cached) were removed from the log, resulting in 297,153 records. This set included all successful external requests, out of these only 173,69 did not originate in an query, i.e. the referral site was a portal, the user typed in the SHIL URL into the location bar or chose it from his/her Favorites list. Note that only 5.8% of the *external hits* did not originate from a search engine. The major source of the *external queries* was google.co.il (71.6%), followed by google.com (10.6%), search.msn.co.il (6.3%) and walla.co.il (a local portal and search engine, 3.6%). Thus, over 80% of the *external queries* originated from Google sites.

The huge majority of the queries were in Hebrew, with some Arabic and Russian, while almost none used Latin characters. The search engines employ various techniques for encoding the non-Latin queries in the referral URL, and it is not always easy or possible to filter out the actual query from this text. For example for queries originating from MSN, the query was passed to the SHIL server and logs as a series of question marks. Some of these problems are caused by the use of several standards: some sites encode Hebrew and Arabic as single byte strings while other sites as multi byte (utf-8) strings. Additional variations in encoding are caused by the fact that Hebrew and Arabic are written from right to left. After filtering out the majority of illegible queries, like queries with question marks only, null queries, non-character strings; the dataset including 266,295 *external queries* was loaded into MS Access, a relational database, for further analysis.

### 4 Results and Discussion

We report basic characteristics of the *external query* logs: length of queries, most frequent queries, most frequent query words, and distribution of requests both as a function of the specific date and as a function of the time of day at the server. The results of our analysis are compared with the data reported for the Utah state government website [3] and with the results of the analyses of the logs of general search engines [16].

4.1 Query Length and the Use of Search Modifiers

Query length counts the number words in the query, where a word is a string of characters delimited either by a space or by the end of the query. Quotation marks were removed from the queries prior to counting the number of words in the query. The results appear in Table 1.

**Table 1.** Query length distribution in absolute numbers and percentages out of the 266,295 queries

| Query length in words | No. Occurrences | % of queries | cumulative % |
|-----------------------|-----------------|--------------|--------------|
| 1 word                | 15,817          | 5.94%        | 5.94%        |
| 2 words               | 113,407         | 42.59%       | 48.53%       |
| 3 words               | 77,736          | 29.19%       | 77.72%       |
| 4 words               | 40,234          | 15.11%       | 92.83%       |
| 5 words               | 12,693          | 4.77%        | 97.59%       |
| 6 words               | 4,275           | 1.61%        | 99.20%       |
| 7 words               | 1,258           | 0.47%        | 99.67%       |
| 8 words               | 491             | 0.18%        | 99.86%       |
| 9 words               | 196             | 0.07%        | 99.93%       |
| 10 words              | 79              | 0.03%        | 99.96%       |
| 11 words              | 32              | 0.01%        | 99.97%       |
| 12 words              | 39              | 0.01%        | 99.99%       |
| more than 12 words    | 44              | 0.02%        | 100.00%      |

The mean number of query length is 2.9, which is comparable with other search engine log analyses ([3, 16, 23]), however the distribution of the query lengths is rather different: in the SHIL log the percentage of single-word queries is surprisingly low as can be seen in Table 2.

In the SHIL on the Web log, 86.9% of the queries are between two and four words long, whereas for the other logs the percentage ranges between 59.7% and 67.8%. This finding is rather surprising, especially since Hebrew is a “compact” language.

**Table 2.** Reported query lengths in several studies, cumulative percentages are in parentheses

| Query length     | SHIL 2005        | Utah 2003        | AlltheWeb 2002   | AltaVista 2002   | AlltheWeb 2001   | Excite 2001      | Knoxville 1997-2001 |
|------------------|------------------|------------------|------------------|------------------|------------------|------------------|---------------------|
| 1                | 5.9%<br>(5.9%)   | 30.7%<br>(30.7%) | 33.1%<br>(33.1%) | 20.4%<br>(20.4%) | 25.1%<br>(25.1%) | 25.0%<br>(25.0%) | 38.8%<br>(38.8%)    |
| 2                | 42.6%<br>(48.5%) | 37.0%<br>(67.7%) | 32.6%<br>(65.7%) | 30.8%<br>(51.2%) | 35.8%<br>(35.8%) | 31.7%<br>(56.7%) | 41.5%<br>(80.3%)    |
| 3                | 29.2%<br>(77.7%) | 19.2%<br>(86.9%) | 18.9%<br>(84.6%) | 22.8%<br>(74.0%) | 22.4%<br>(83.3%) | 22.8%<br>(79.5%) | 13.4%<br>(93.7%)    |
| 4                | 15.1%<br>(92.8%) | 7.6%<br>(94.5%)  | 8.2%<br>(92.8%)  | 12.0%<br>(86.0%) | 9.6%<br>(92.9%)  | 8.8%<br>(88.3%)  | 6.2%<br>(99.9%)     |
| mean term length | 2.79             | 2.25             | 2.30             | 2.92             | 2.40             | 2.60             | 2.0                 |

Most of the prepositions, the definite article and some of the conjunctions are prefixed to the word and are not stand-alone words, for example The Ministry of Finance, is only a two-word phrase in Hebrew – MISRAD HAOTZAR. Thus we expected that in the SHIL log short queries will be more prevalent than in the mainly English logs.

One possible explanation is that users reach the SHIL Website having rather complex problems, and the phrasing of such problems requires longer queries. If we accept this explanation, we should expect a similar finding for the Utah government website. However, the query length distribution for the Utah site does not differ from the distribution for the general search engines. The longest query for the Utah government website was a 40-word query, for SHIL the longest query contained 56 words (due to some automatic transformation by the search engine), the longest “real” query was 40 words long for SHIL on the Web as well.

Our definition of a query word includes Boolean operators (e.g. AND, I, +, -). We Note, that the plus and minus signs, when used properly, i.e. to force inclusion or exclusion of a certain query term, should not be separated from the query word, however there were multiple occurrences to space delimited– signs (1,434 queries with space delimited – signs). We cannot say anything conclusive about the use of the + sign, since it is also used by the search engines to encode spaces. The number of queries in which minus signs was used properly (placed next to the query word without a space between them) was only 183 (0.07% of the queries). Inspecting the queries including space delimited minus signs, it seems that in most case the minus sign was not introduced to exclude a certain term, but as a delimiter, e.g. Ministry of Transport – fines. AND, OR and I appeared in 26, 420 and 180 queries respectively. The most frequently used query modifier was the quotation marks for phrase search (7,617 queries, 2.9% of the queries). Jansen, Spink and Saracevic [9] in their analysis of an Excite log of about 50,000 queries, found that Boolean operators and query modifiers (+/-) were used in 24.7% of the queries, and were misused in about 34.9% of the cases. In our query log, the occurrence of Boolean operators and search modifiers was only 3.6%, however the percentage of misuse was low - 1.9% (assuming all the phrase searches were constructed properly).

## 4.2 Most Frequent Queries and Query Words

The query log included 266,295 queries. Of these, 72,799 unique queries were identified. In Table 3 the twenty five most frequent queries and their meanings in English are displayed. All the above queries are Hebrew queries, the most frequently occurring query in Arabic was تعلم اللغة العبرية (learning the Hebrew language) which occurred 286 times in the log. The most frequently occurring Russian query, социальное обеспечение (social security), occurred only 22 times. SHIL on the Web is currently being translated to Russian, and at the time the log was collected only a minority of the information in Hebrew was available in Russian as well. In the query log there were 2,729 Arabic queries (1.02%), 614 queries in Russian (0.23%) and 1162 queries included Latin characters (0.44%).



**Table 3.** Most frequently occurring queries and their meanings, in absolute numbers and in percentages (N=266295). Text in *italics* was added to clarify the query.

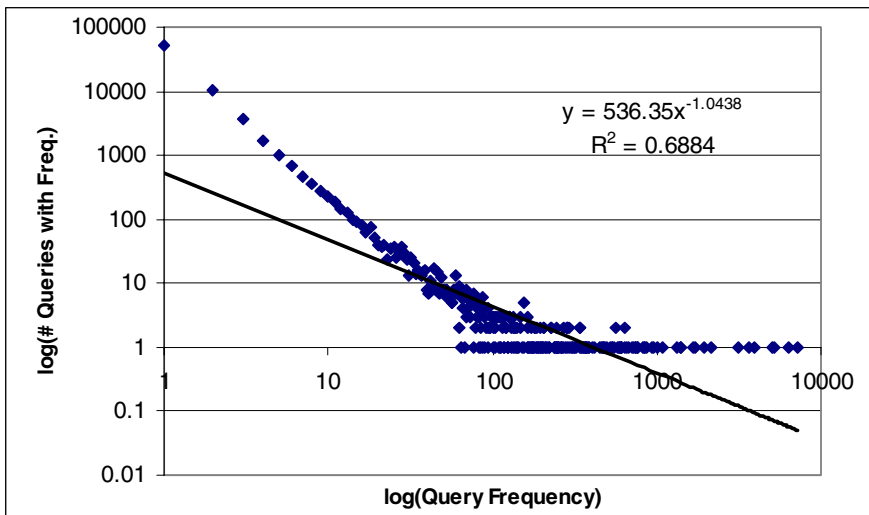
| query in original language     | query in English   | Freq. | %     |
|--------------------------------|--|-------|-------|
| ביטוח לאומי                    | National Insurance   | 7226  | 2.71% |
| משרד הפנים                     | Ministry of Internal Affairs   | 6394  | 2.40% |
| משרד העבודה                    | Ministry of Social Affairs   | 5234  | 1.97% |
| שעון קיץ                       | daylight saving time   | 5044  | 1.89% |
| חוזת שכירות                    | rental contract  | 3891  | 1.46% |
| משרד הרישוי                    | licensing authority  | 3637  | 1.37% |
| סופרלנד                        | Superland ( <i>amusement park</i> )                                    | 3160  | 1.19% |
| בטוח לאומי                     | National Insurance ( <i>variant spelling</i> )                         | 2119  | 0.80% |
| דמי הבראה                      | vacation fees  | 1933  | 0.73% |
| פיצויי פיטורין                 | dismissal compensation   | 1728  | 0.65% |
| משרד הרישוי חולון              | licensing authority Holon ( <i>the central office is in Holon</i> )    | 1685  | 0.63% |
| בית משפט לתביעות קטנות         | Small claims court   | 1392  | 0.52% |
| בית החייל                      | Soldiers' residence  | 1383  | 0.52% |
| היחידה להכוונת חיילים משוחררים | The unit for advising discharged soldiers                              | 1327  | 0.50% |
| רשם העמותות                    | Non-profit organizations' registrar                                    | 1072  | 0.40% |
| ש.י.ל.                         | SHIL   | 994   | 0.37% |
| חוק הגנת הצרכן                 | consumer protection law  | 968   | 0.36% |
| חוק הגנת הדייר                 | tenant protection law  | 941   | 0.35% |
| מינהל מקרקעי ישראל             | Israel Land Administration   | 867   | 0.33% |
| חיילים משוחררים                | discharged soldiers  | 854   | 0.32% |
| האגודה לתרבות הדיור            | Housing Advice Association   | 848   | 0.32% |
| קו לחיים                       | Kav LaHaim ( <i>an organization providing help for sick children</i> ) | 830   | 0.31% |
| משרד רישוי                     | Licencing Authority ( <i>variant spelling</i> )                        | 763   | 0.29% |
| לשכת הבריאות                   | Ministry of Health   | 753   | 0.28% |
| משרד העבודה והרווחה            | Ministry of Social Affairs ( <i>full name of the Ministry</i> )        | 750   | 0.28% |

Let us consider the most frequently occurring query, National Insurance. The users were looking for the National Insurance Institute of Israel ([http://www.btl.gov.il/English/eng\\_index.asp](http://www.btl.gov.il/English/eng_index.asp)). The query appeared in several variants, two of them appear among the top-twenty queries. By inspecting the frequently occurring queries, we came across a few more variation, altogether SHIL was queried for the National Insurance Institute at least 10,326 times (not counting queries where the users were looking for specific offices) – 3.9% of the queries. When submitting the query National Insurance (in Hebrew) to Google, the top result, of course, is the National Institute's site. The SHIL result, as of January 17, 2006, comes out number four for the first variant (with a \*) and number two for the second variant. We have no information how many users chose the National Insurance site, but a considerable number of users preferred to look at the information provided by SHIL on the topic. A possible reason for this could be that the snippet Google displays for the SHIL page on the National Insurance Institute

summarizes the page content in a much more appealing fashion. The SHIL snippet states: “National Insurance – all the information about you rights ...”; while the National Insurance Institute’s snippet says: “Notice to the employees about changes in the insurance fees ...”.

The SHIL page is the top result for “daylight saving time” (in Hebrew) on Google as of January 17, 2006. This is an example of a query where it is rather unclear in advance which site can provide an answer. We presume the users were interested about the dates Israel switched to and from the daylight saving time. In Israel, the settings of daylight saving time changed frequently, as it was the topic of political haggling. Currently the SHIL page on the Superland is only the seventh result on Google (could have been much higher during the time the data was collected). Seemingly the amusement park does not have a homepage of its own, thus people looking for information on the park or on a raging controversy regarding its admission policies for disabled children have to turn to other sites.

Comparing the top queries in the SHIL log to the top queries in the Utah state website [3], we see some similarities. Frequently occurring queries the Utah log included: employment, unemployment, jobs (queries similar to “dismissal compensation”, “Ministry of Social Affairs”), real estate (somewhat similar to “tenant protection law”, “Housing Advice Association”), drivers license (similar to “Licensing Authority”) and Medicaid (“Ministry of Health”).



**Fig. 2.** Rank-frequency distribution of queries

Figure 2 depicts the rank-frequency distribution of the queries on a log-log scale. There were a few frequently occurring queries, but the majority of the queries (52,065, 71.5% of the unique queries) occurred only once.

We also looked at the frequencies of the query terms. Altogether 742,454 query terms were extracted from the 266,295 queries. The number of unique query terms was

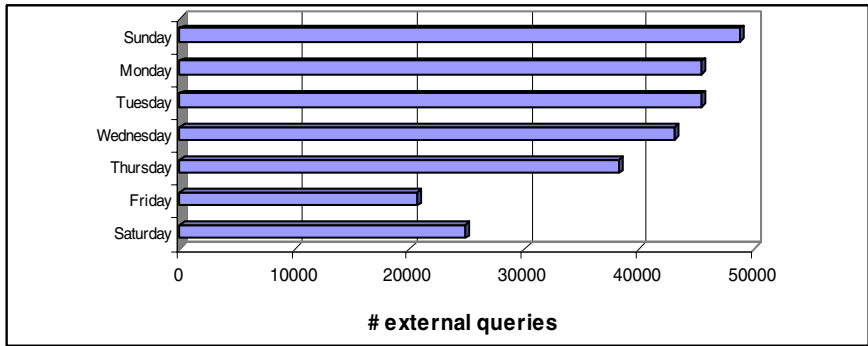
18,845. In Table 4 the twenty-five most frequently occurring query terms with their translation are displayed. Most of these words are parts of the most frequently occurring queries, see Table 3. As we said before, prepositions, the definite article and the possessive form are pre and postfixed to the word. We have not stemmed the query terms, not only because of the complexity of the process, but because the currently available large, commercial search engines do not apply morphological analysis to Hebrew or to Arabic.

**Table 4.** Top 25 query terms and their translations in absolute numbers and in percentages out of the total terms (N=742,454)

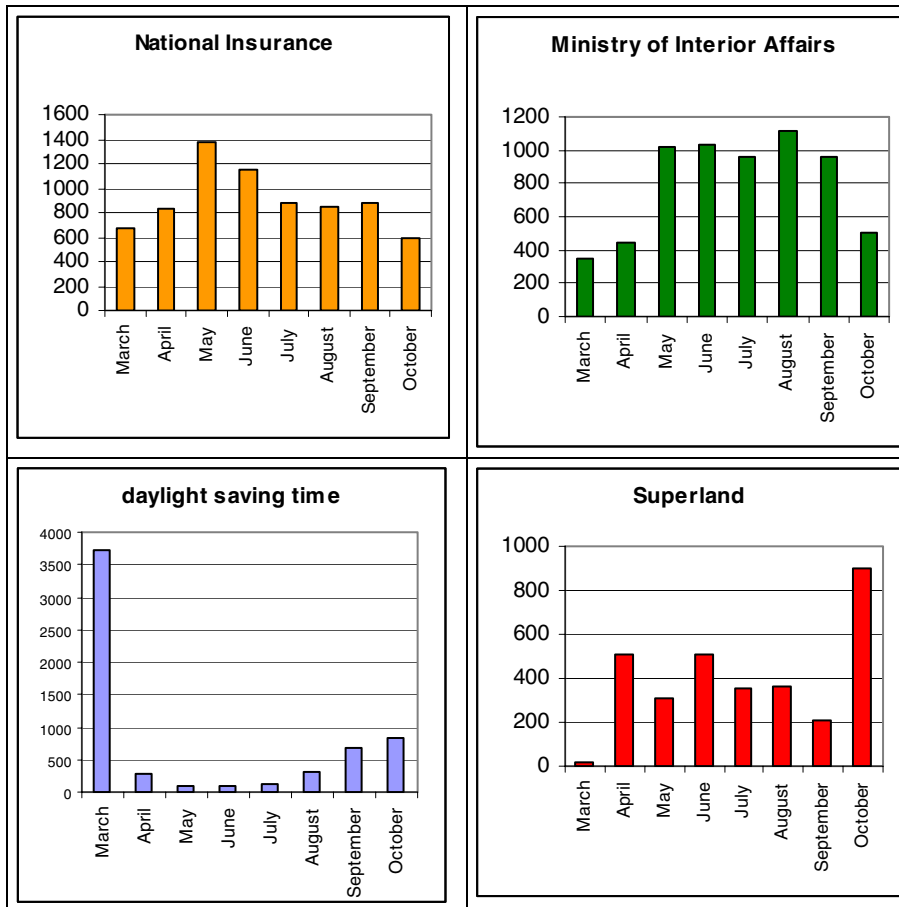
| original term | translated term         | no. occurrences | % of terms |
|---------------|-------------------------|-----------------|------------|
| משרד          | ministry, office        | 36778           | 4.95%      |
| לאומי         | national                | 19973           | 2.69%      |
| ביטוח         | insurance               | 15496           | 2.09%      |
| בית           | home, house             | 13462           | 1.81%      |
| חוק           | law                     | 13430           | 1.81%      |
| הפנים         | the interior            | 10184           | 1.37%      |
| העבודה        | the work                | 9181            | 1.24%      |
| הרישוי        | the licensing           | 8675            | 1.17%      |
| עבודה         | work                    | 7318            | 0.99%      |
| מס            | tax                     | 7102            | 0.96%      |
| דמי           | fee                     | 6256            | 0.84%      |
| שעון          | clock                   | 6188            | 0.83%      |
| קיץ           | summer                  | 6139            | 0.83%      |
| חוזה          | contract                | 5558            | 0.75%      |
| שכירות        | renting                 | 5268            | 0.71%      |
| הכנסה         | income                  | 4934            | 0.66%      |
| הבראה         | vacation, convalescence | 4620            | 0.62%      |
| לשכת          | bureau of               | 4584            | 0.62%      |
| פיטורין       | dismissal               | 4386            | 0.59%      |
| פיצוי         | compensation            | 3914            | 0.53%      |
| זכויות        | rights                  | 3906            | 0.53%      |
| חיילים        | soldiers                | 3837            | 0.52%      |
| סופרלנד       | Superland               | 3781            | 0.51%      |
| משוחררים      | discharged              | 3765            | 0.51%      |
| רשם           | registrar               | 3704            | 0.50%      |

4.3 Seasonal and Daily Effects

The number of external query requests per month varied between 22,232 (March) and 38,466 (May). The daily distribution of external queries is displayed in Figure 3. The busiest day of the week is Sunday (a workday in Israel). It is interesting to note that there are more requests on Saturdays (weekend) than on Fridays, which is a partial workday in Israel. The busiest hours in the day are between 10:00AM and 1:00PM, over 50% of the requests arrive between 9:00 AM and 5:00PM, but there are requests at all hours of the day and night – the slowest hour is between 4:00AM and 5:00AM.



**Fig. 3.** The distribution of the external queries over the days of the week



**Fig. 4.** Seasonal effects on a sample of queries

We also examined the time distribution of some of the more popular queries. The query “daylight savings time” was highly seasonal, the peak months for this query were March (when Israel switches to summer time) and again in September-October (when Israel switches back to winter time). In Israel, the exact date of switching from summer to winter time and back changes yearly due to the Jewish holidays. The query Superland (amusement park) was also expected to be seasonal, this time the expected peak months were July and August (summer vacation), but the results show that April and October were strong, probably because of the Jewish holidays during those months, and in reaction to the press coverage mentioned earlier. For the queries “National Insurance” and “Ministry of Internal Affairs” there were only moderate monthly fluctuations. The monthly distributions for these queries appear in Figure 4.

## 5 Conclusions

SHIL on the Web is a large content site that provides advice to citizens. It seems that awareness of this site among Israeli Internet users was not very high as the number of hits is very high, but most are initiated through search engines. Thus it seems that SHIL serves the public well even without brand awareness. In the future we plan to incorporate the log analysis with user studies in order to gain a better understanding of users’ citizenship-related information needs.

## References

1. Beitzel, S. M., Jensen, E.C., Chowdhury, A., Grossman, D., Frieder, O.: Hourly analysis of a very large topically categorized Web query log. In: SIGIR’04 (2004) 321-328.
2. Cacheda, F. Vina, A.: Experiences retrieving information in the World Wide Web. In ISCC’01 (2001) 72-79.
3. Chau, M., Fang, X., Sheng, O. R. L.: Analysis of the query logs of a Web site search engine. *Journal of the American Society for Information Science and Technology*, 56 (2005) 1363-1376.
4. Croft, W., Cook, R., Wilder, D.: Providing government information on the Internet: Experiences with THOMAS. Paper presented at the Digital Libraries Conference, Austin, TX. (1995) <http://thomas.loc.gov/home/dlpaper.html>
5. Davis, P. M.: Information-seeking behavior of chemists: A transaction log analysis of referral URLs. *Journal of the American Society for Information Science and Technology*, 55 (2004) 336-332
6. Jansen, B. J., Spink, A.: An analysis of Web searching by European AlltheWeb.com users. *Information Processing and Management*, 41 (2005) 361-381.
7. Jansen, B. J., Spink, A.: How are we searching the World Wide Web? A comparison of nine search engine transaction logs. *Information Processing and Management*, 42 (2006) 248-263.
8. Jansen, B. J., Spink, A., Pedersen, J.: A temporal comparison of AltaVista Web searching. *Journal of the American Society for Information Science and Technology*, 56 (2005) 559-570.
9. Jansen, B. J., Spink, A., Saracevic, T.: Real life, real users and real needs: A study and analysis of user queries on the Web. *Information Processing and Management*, 36 (2000), 207-227.

10. Jones, S., Cunningham, S.J., McNam, R.: Usage analysis of a digital library. In *Proceedings of the Third ACM Conference on Digital Libraries* (1998) 293–294.
11. Novak, T. P., Hoffman, D. L.: New metrics for new media. *W3C Journal*, 2 (1997), <http://www.w3journal.com/5/s3.novak.html>
12. Ozmutlu, S., Spink, A., Ozmutlu, H. C.: A day in the life of Web searching: An exploratory study. *Information Processing and Management*, 40 (2004) 319–345.
13. Park, S., Lee, J. H., Bae, H. J.: End user searching: A Web log analysis of NAVER, a Korean Web search engine. *Library and Information Science Research*, 27(2005), 223–231
14. Ross, N. C. M., Wolfram, D.: End user searching on the Internet: An analysis of term pair topics submitted to the Excite search engine. *Journal of the American Society for Information Science*, 51 (2000), 949–958.
15. Silverstein, C., Henzinger, M., Marais, H., Moricz, M.: Analysis of a very large Web search engine query log. *ACM SIGIR Forum*, 33(1). <http://www.acm.org/sigir/forum/F99/Silverstein.pdf> (1999)
16. Spink, A. Jansen, B. J.: *Web search: Public searching of the Web*. Kluwer Academic Publishers, Dordrecht (2004).
17. Spink, A., Jansen, B. J., Wolfram, D., Saracevic, T.: From e-sex to e-commerce: Web search changes. *IEEE Computer*, 35 (2002), 107–109.
18. Spink, A., Ozmutlu, S., Ozmutlu, H. C., & Jansen, B. J. : U.S. versus European Web searching trends. *SIGIR Forum*, Fall 2002, <http://www.acm.org/sigir/forum/F2002/spink.pdf>
19. Spink, A., Wolfram, D., Jansen, M. B. J., Saracevic, T.: Searching the Web: The public and their queries. *Journal of the American Society for Information Science and Technology*, 52 (2001), 226–234.
20. Thelwall, M.: Web log file analysis: backlinks and queries. *Aslib Proceedings*, 53 (2001) 217–223.
21. Wikipedia.: Referer. 2006. <http://en.wikipedia.org/wiki/Referer>
22. Wang, P., Berry, M. W., Yang, Y.: Mining longitudinal Web queries: Trends and patterns. *Journal of the American Society for Information Science and Technology*, 54 (2003), 743–758
23. Wolfram, D., Spink, A., Jansen, B. J., Saracevic, T.: Vox populi: The public searching of the Web. *Journal of the American Society for Information Science and Technology*, 52 (2001), 1073–1074.
24. Xie Y., O'Hallaron, D: Locality in search engine queries and its implications for caching. In *Proceedings of INFOCOM'02* (2002) 307–317.

# On Mediated Search of the United States Holocaust Memorial Museum Data

Jefferson Heard\*, Jordan Wilberding\*, Gideon Frieder\*\*,  
Ophir Frieder, David Grossman, and Larry Kane\*

Information Retrieval Lab  
Illinois Institute of Technology

**Abstract.** The United States Holocaust Memorial Museum (USHMM) recently celebrated its ten-year anniversary. The museum was established to bear witness to the human atrocities committed by the Nazi reign of terror. As such, related data must be collected, and means to store, search, and analyze the data must be provided. Presently, the data available reside in various formats, sizes, and structures, in videotape and films, in microfilms and microfiche, in various incompatible structured databases, as unstructured electronic documents, and semi-structured indexes scattered throughout the organizations. Collected data are partitioned over more than a dozen languages, further complicating their processing. There is currently no single search mechanism or even department of human experts that can sift through all the data in a fashion that provides global, uniform access. We are currently experimenting with our developed Intranet Mediator technology to provide answers, rather than a potential list of sources as provided by common search engines, to questions posed in natural language by Holocaust researchers. A description of a prototype that uses a subset of the data available within the USHMM is described.

## 1 Introduction

Quoted directly from its website, the mission of the United States Holocaust Memorial Museum (USHMM) states:

“The United States Holocaust Memorial Museum is America’s national institution for the documentation, study, and interpretation of Holocaust history, and serves as this country’s memorial to the millions of people murdered during the Holocaust.”

We are developing a Mediator search tool to provide a natural language, single point of querying interface to the United States Holocaust Memorial Museum data in all its various formats. This effort is conducted independently from, but in cooperation with the archives division of the USHMM.

---

\* Also of Intranet Mediator Inc.

\*\* Also of George Washington University.

The Intranet Mediator [3] is a search tool based on patented technology [2] capable of providing a layer of abstraction over structured (SQL) databases, semi-structured (XML, MARC, LOC) document repositories, and unstructured text repositories using their own search engines. This layer of abstraction allows the user a natural-language interface to query all of these repositories in an intelligent manner from a single place. It also places no requirements on the underlying systems that it searches. We have developed a system that allows a researcher to enter a natural language question and query up to six sources, some of which are structured, semi-structured (XML), and text, all at the direction of the Mediator.

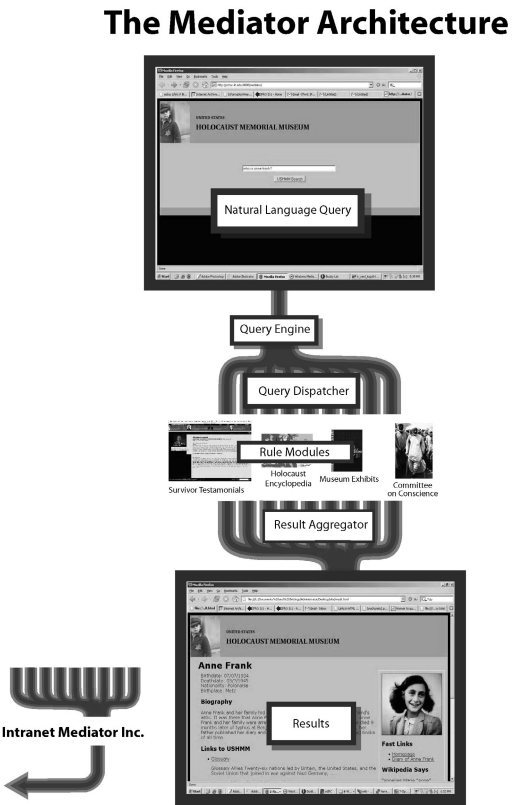


Fig. 1. Architecture of the Intranet Mediator

The USHMM’s archives contain many collections in languages other than English. In addition to our work in this study, we are developing a method of querying these collections in their own languages and returning interpretable results. Towards this, we have included in our prototype an English search over the Slovak Jewish Census collection, a small index of boxes of text and film documents that contains abstracts written in the Slovak language of each record.



In the search of these documents, results are presented to the user in their original language of Slovak. For description on recent efforts in cross language information retrieval, see [4].

## 2 Related Work

Question and Answer systems abound and include the likes of START [7] and JAVELIN [9]. These systems handle natural language questions, like our Mediator, however they operate on only one kind of data and most often use “canned” answers, such as Wikipedia for their results. In addition, the Mediator uses techniques from Natural Language Processing, such as named-entity extraction [8] and text classification [1] to extract the necessary data from the queries to pose them to the different types of systems and also to direct them towards the most salient procedure for obtaining and composing a final answer.

## 3 Solution Architecture

The global architecture of the Intranet Mediator is illustrated in Figure 1. When a user issues a natural language question to the Mediator, a *Redirector* routes it through several levels of processing and then directs the user’s browser to the answer. Inside the *QueryEngine*, the user’s query is first broken up by a *Tokenizer*. The sequence of tokens is then analyzed by a *KnownEntityTagger* that finds, concatenates, and tags subsequences that exist within the structured repositories available to the Mediator. This sequence of tagged data is passed through an *InferredEntityTagger*, which further tags data through finding patterns that have previously surrounded a known entity. The *KnownEntityTagger* finds entities in the query by matching substrings with the results of one or more pre-defined structured queries on the Mediator accessible databases. The *InferredEntityTagger* takes the results from the *KnownEntityTagger* and uses them as training as well as checks them for inferrable entities. The *InferredEntityTagger* uses a simple trigram tagging scheme similar to [6] to find entities.

This digested query is sent to the *QueryDispatcher*, which passes the query through a *BayesianLogisticRegressionClassifier*, selecting the top-level rule with the best potential to answer the question. The sequence of tagged entities and the name of the selected rule are then hashed together to determine if the answer has already been cached. If the question has no cached answer, the top-level rule is given the query and entity values and the *QueryDispatcher* executes it. A top-level rule, shown in Figure 2, consists of:

- A set of questions with known answers, found in the *<questions>* section which is used to train the classifier.
- A set of required fields, found in the *<required-fields>* section which is filled in by the entities found in the query and by sub-rules that use the top-level rule’s information to spelunk for more data. Assigned along with each field is the probability the top-level rule can answer the query given that the field cannot be provided by either the query or a sub-rule.

```

<rule name='Person'>
  <questions recognizer='com.imi.mediator.querydispatcher.BLRQueryRecognizer'>
    <query>Who was $Name?</query>
    <query>Who is $Name?</query>
    ...
  </questions>
  <required-fields>
    <field p='0.0' name='Name' />
    <field p='0.8' name='Birthdate' />
    <field p='0.8' name='DeathDate' />
    ...
  </required-fields>
  <logic>
    required_fields['Name'] = titlecase(required_fields['Name'])
  </logic>
  <answer>
    <subanswer fields='Name'><h1>%</h1><hr /></subanswer>
    <subanswer fields='Birthdate,Deathdate'><h2>(%-%)</h2></subanswer>
    ...
  </answer>
</rule>

```

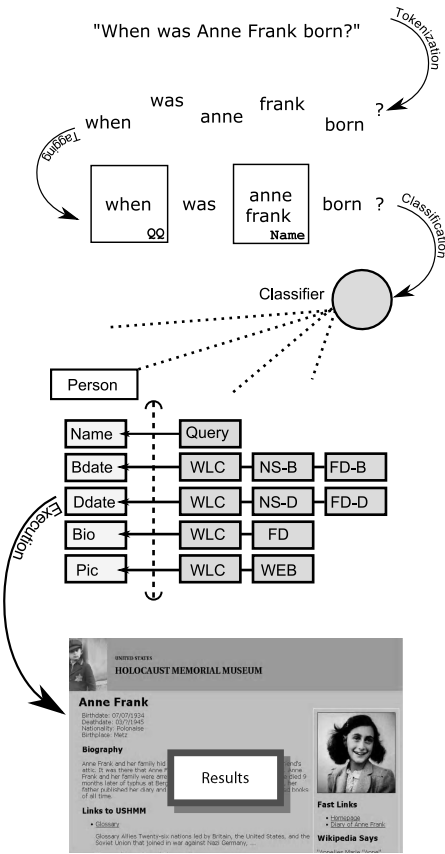
**Fig. 2.** A sample rule

- A logic section, denoted as *<logic>*, written in Python [10] that operates via *QueryModules* with the searchable data repositories and the set of fields filled in by the query.
- A set of formatting instructions for the *Aggregator*, the *<answer>* section that take various pieces of the result data and make them human-readable.

A rule, in our approach, is a template that answers an entire class of similarly phrased questions. Once a rule is selected and query fields are filled in, sub-rules are selected to create an execution plan. A sub-rule is a rule that contains no set of answerable questions, but merely a list of fields the rule requires to execute and a list of fields the rule provides upon completion. For each required field in the top-level rule, a priority queue of sub-rules to execute is created. This structure is prioritized by the probability that the sub-rule will assign the correct value to the field. Once the execution plan is created, the first item of each priority queue is polled, and the logic section is executed. Note that the rules are independent from one another and are executed in parallel. Each time a field is successfully filled in by a rule, the field's priority queue is emptied. As long as there are non-empty priority queues, the execution plan continues. As each field is filled, it is passed to the *Aggregator*, which uses the formatting instructions in the top-level rule to build the answer that is displayed to the user.

One of the unique features of the Mediator is that it attempts to return an answer to a question rather than a set of hyperlinks or some other form of a ranked result set, as is common for traditional search engines. While question answering systems abound, no question answering system exists currently that

is capable of exploiting multiple types of data as the IIT Intranet Mediator is [3]. Furthermore, the Mediator is actually querying the underlying data as opposed to selecting from a predetermined set of possible answers as in efforts such as Ask Jeeves (ask.com).



**Fig. 3.** Flow of "When was Anne Frank Born?" through the mediator

An answer is determined entirely by the top-level rule that a question triggers; therefore it is important that the proper rules exist in the system for each question. In general, one top-level rule should exist for each different type of answer needed. In the case of our prototype system, we created a rule for answering questions about people, one for answering questions about places, and one for events. In a more robust system, these might be broken into more subtle rules, such as answers for questions about a person's biography versus questions about a person's geneology, or questions about concentration camps versus questions about a city. All questions are answered by the Mediator; it is simply a matter of granularity of answer that one needs to consider in designing and implementing rules.

To demonstrate the Mediator, in Figure 3, we illustrate the flow of “When was Anne Frank born?” through the system we developed using the USHMM data. The query is first tokenized, case-folded, and then tagged; since “Anne Frank” exists in the USHMM’s structured repositories, this tagging is done via the *KnownEntityTagger*. Then the tagged sequence is passed to the classifier, which decides on “Person” as the rule meaning a person’s biographical data. An execution plan is created for the person rule, including data from four separate sources: the query, the USHMM’s database of victim and survivor names, a.k.a. “NameSearch”, our fact database, and the USHMM web search. The leftmost rule in each queue is executed until there are values filled in for all fields: *Name*, *Bdate*, *Ddate*, *Bio*, and *Pic*. Each final field value will become part of the final answer, shown at the bottom of the figure. The answer resembles a hand-written encyclopedia article despite the fact that it has been assembled from multiple sources on the fly through the work of the Mediator.

Previously, although many retrieval engines were capable of querying multiple sources, they were limited to a single type of data: structured, unstructured, or semi-structured. The Mediator is needed to provide this one-query, one-result approach to managing user requests over these different kinds of data, because queries on each are different both in terms of the method in which a data source is queried and in terms of the nature of the returned results. For example, it may be difficult to “rank” structured results, as all results of a structured query may provide part of an answer or be used in a statistical fashion to provide a more high-level answer based on low level data. However, on an unstructured text repository, ranking of whole documents may be the only way to return data. A single query will behave in vastly different ways depending upon what kind of data it is querying. Mediation is a way of intelligently managing these different behaviors. Our approach to this is that each different type of source is handled by a query module: a highly configurable object that connects to a data source, issues a query, and returns results in a uniform way. In our Mediator, we have implemented a JDBC query module to handle SQL queries over structured data, a CGI query module to handle queries to any web-based CGI script via HTTP GET or POST requests, an XPath query module to handle semi-structured queries against XML repositories, and Google, Yahoo, and AIRE[5] modules to handle unstructured searches over text collections.

To compose an answer from the results returned through these multiple types of queries on multiple sources, we employ a final processing engine called an *Aggregator*. The *Aggregator* takes each field provided and decides how to display it. When there is duplicate or conflicting data this is handled by dropping all duplication except the data with the highest probability of correctness and displaying only that. The Aggregator will display all fields that have confidences above a configured threshold. A simple printf-like XML template language is included as part of a toplevel rule definition which determines the final formatting of displayed results.

Our solution consists of the Mediator running on a single HP DL-380 running Gentoo GNU/Linux 2005.1 with access to data sources residing on the local

machine, other machines on the LAN, and at the USHMM. We have rules for answering questions about people, places, events, and a default rule that answers unknown queries with a set of hyperlinks via a traditional search engine. To date, six data sources are incorporated into the answers we provide:

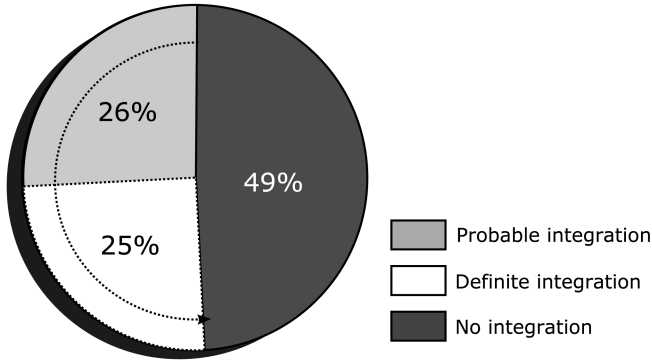
- USHMM web-site text search
- Learning Center Database: an MS SQL Server database with the full content of the USHMM Permanent Exhibit, including pictures, videos with their subtitles in French, Spanish, and English, text snippets with structure associated, name databases, and more
- Hand-built database of facts, including birthdates and deathdates and some small amount of biographical data on famous people from World War II.
- USHMM NameSearch, a CGI-interfaced database that resides on the web-site and contains thousands of names of survivors and victims and their associated data.
- Index of the archives of the ministry of interior of the Slovak government in the years 1938-1945 in the Slovak language.
- Semi-structured location database of townships, metropolitan areas, and counties in Slovakia that had Jewish residents before the Holocaust.

**Table 1.** Frequency of most common queries

| Query               | Occurrences | Percent of Log |
|---------------------|-------------|----------------|
| Holocaust           | 6409        | 2.37%          |
| Auschwitz           | 5190        | 1.92%          |
| Anne Frank          | 4799        | 1.78%          |
| Concentration Camps | 3753        | 1.32%          |
| Hitler              | 2934        | 1.09%          |

We obtained from the USHMM a query log providing all user searches over an six-month period of time, including major Jewish holidays, for a total of approximately 270,000 queries. A team of researchers analyzed these queries for uniqueness and total number of occurrences within the set, and we set out to cover as many of the most common queries as possible. We treat queries referring to the same named entity, such as "Adolph Hitler" and "Hitler" as equivalent, since our answer should be the same in both cases. In Table 1, we illustrate the five most common queries and their frequencies across the log. We created several query types and tested these against a sample of the query log to determine the coverage of queries that were handled in some manner more intelligently than simply passing all the query terms along to the USHMM web search engine. That is, we determined the coverage of queries that experienced some integration of data across several sources.

To analyze our system's coverage of the query log, we first sorted the query log by the number of occurrences of each query, then took the top 50% of the queries in the sorted list. These queries were issued to the Mediator and 50%



**Fig. 4.** Percentage of queries in log benefiting from mediation

reported some form of data integration beyond the unstructured USHMM web search. In Figure 4, these queries form the pie slice designated “definite integration”. After we had confidence in these results, we took a random sample of one thousand queries throughout the whole query log and issued these to the mediator, with a resulting 51% of these queries returning results integrated from more than the unstructured USHMM web search. In Figure 4, the additional handled queries implied by our statistical sample form the pie slice designated “probable integration”. The pie slice labeled “no integration” covers only those queries that would return answers containing only links from the web search.

## 4 Future Work

We would like to develop a way to analyze query data so that likely rules can be automatically suggested, if not added to the system. As we added more rules to our engine, the number of mediation-enhanced queries grew rapidly. We do not yet know the trade-off point where adding new rules does not significantly increase the coverage our engine gets. As more sources were made available to the engine, its answers became more accurate and better focused. We are working to determine a metric for how to define “accurate” so that we can measure the amount of contribution having a particular data source available makes to the overall answer.

## Acknowledgments

We gratefully acknowledge the cooperation of United States Holocaust Memorial Museum in providing access to all data used herein. We also acknowledge the Holocaust IPRO team from the Illinois Institute of Technology. Special thanks go to Steven Beitzel, Eric Jensen, and Michael Lee, the primary architects and

developers of the initial Intranet Mediator[3]. We gratefully acknowledge the Intranet Mediator, Inc. for its generous support. This effort is dedicated to the millions who lost their lives in the Holocaust. HY”D.

## References

1. S. Eyheramendyl, A. Genkin, W. Ju, D. Lewis and D. Madigan. Sparse Bayesian Classifiers for Text Categorization. *Journal of Intelligence Community Research and Development*, Volume 13, 2003.
2. O. Frieder and D. Grossman. Intranet Mediator. US Patent #6,904,428, June 2005.
3. D. Grossman, S. Beitzel, E. Jensen, and O. Frieder. IIT Intranet Mediator: Bringing data together on a corporate intranet. *IEEE IT PRO*, January/February 2002.
4. D. Grossman and O. Frieder. *Information Retrieval: Algorithms and Heuristics*. Springer Publishers, 2nd ed., 2004.
5. T. Infantes-Morris, P. Bernhard, K. Fox, G. Faulkner, K. Stripling. Industrial Evaluation of a Highly-Accurate Academic IR System. *ACM CIKM*, New Orleans, Louisiana, 2003.
6. D. Jurafsky and J. Martin. *Speech and Language Processing*. pp. 577-583. Prentice Hall, 2000.
7. B. Katz, G. Marton, G. Borchardt, A. Brownell, S. Felshin, D. Loreto, J. Louis-Rosenberg, B. Lu, F. Mora, S. Stiller, O. Uzuner and A. Wilco. External Knowledge Sources for Question Answering. In the *Proceedings of TREC-2005*, Gaithersburg, Maryland, November 2005.
8. C. Manning and H. Schutze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999, pp.353
9. E. Nyberg, R. Frederking, T. Mitamura, M. Bilotti, K. Hannan, L. Hiyakumoto, J. Ko, F. Lin, L. Lita, V. Pedro and A. Schlaikjer. JAVELIN I and II Systems at TREC 2005. In the *Proceedings of TREC-2005*, Gaithersburg, Maryland, 2005.
10. The Python Language. <http://www.python.org>.

# A Repository of Services for the Government to Businesses Relationship

Daniele Barone<sup>1</sup>, Gianluigi Viscusi<sup>1</sup>, Carlo Batini<sup>1</sup>, and Paolo Naggari<sup>2,\*</sup>

<sup>1</sup> Dipartimento di Informatica Sistemistica e Comunicazione (Disco)

Università degli Studi di Milano-Bicocca - Italy

daniele.barone@unimib.it, gianluigi.viscusi@unimib.it,  
batini@disco.unimib.it

<sup>2</sup> Gruppo CM - via Simone Martini, 126 - 00142 - Roma - Italy

paolo.naggari@gruppcm.it

**Abstract.** We describe an experience of design of a repository of services, performed in an Italian project called Government for Businesses (G4B). The repository G4B represents several properties and characteristics of a wide number of services provided by public administrations to businesses in Italy. An ontology has been built on top of the repository, enriching it with several unifying classifications and metadata. The different usefulness of the repository for service providers, public administrations, businesses are discussed, and several examples are presented.

## 1 Introduction

This paper describes the main characteristics and results of a project, called Government for Business (in short G4B), granted by the Italian Ministry of Industry in years 2003-2005. The ownership of the project is of the CM Group, an Italian company that in the last years undertook several projects involving innovative ITC services and solutions. The project G4B aims at building a technological infrastructure to enable the businesses to make effective use of public administration services. In their life cycle, businesses have to interact with several agencies to request administrative services. The interactions are needed in correspondence of several types of business life events (e.g. starting a new business, evolving a business, etc.). In the G4B context, the focus of the research activity has been, initially, to build a repository of services to businesses with the goal of enabling the CM Group to plan the priorities in the development and selling of services to businesses in the Italian market. As long as the repository design went on, it became more and more clear that the repository could represent a knowledge base that can be useful, not only for the CM Group, but also for the two major players of the G2B relationship, namely public administration and businesses. In the present work we define the motivations and the various conceptual and architectural choices underlying the design of the repository. The paper is organized as follows. In Section 2 we

---

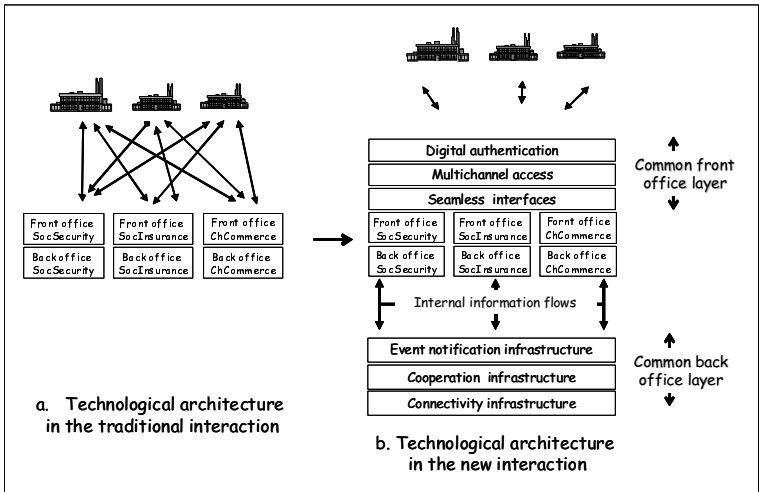
\* This work is an outcome of two years of tight collaboration with the management and personnel of the CM Group (Italy). Special thanks are due to Enza Bruno Bossio and all the staff of the G4B project.



introduce the motivations for using the repository and the evolution of the technological architectures in e-Government interactions. Section 3 introduces the repository and discusses its role in the architecture. In Section 4 we describe the ontology built on top of the repository. Sections 5, 6, and 7 discuss the different utilities that the repository may provide respectively to service developers and providers, public administrations, and businesses. Several examples are provided. Sections 8 and 9 on related and future work conclude the paper.

## 2 New ICT Architectures Adopted for the Government to Business Interactions

In all countries, in the past few years, in order to improve the quality of services for citizens and businesses in their interaction with public administration, many projects have been set up that make effective use of information and communication technologies (ICT). In the present work we focus in particular on the Italian experience and on the supplying of services to businesses. In the past, the lack of cooperation between the administrations led to the establishment of heterogeneous and isolated systems. As a result, two main problems have arisen, i.e. duplicated and inconsistent information and “stovepipe” provision of services. In the old-fashioned style, the interaction between agencies and businesses involves multiple transactions against the agencies’ proprietary interfaces. Moreover, in the interaction with businesses, agencies manage both agency-specific information about them, such as e.g. tax reports, balance sheets, and information that is common to all of them, including identifiers, headquarters and branches addresses, legal form. One major consequence of having multiple disconnected views for the same information, is that businesses experience severe service degradation during their interaction with the agencies.



**Fig. 1.** New technological architecture for Government to Business interactions

Taking into account these issues, the approach followed in order to improve the interaction between administrations and to make more effective service provision, is based typically on a cooperative architecture that, with some variants, follows the general structure shown in Figure 1, where in the left hand side the traditional stovepipe architecture used in the interactions among businesses and administrations is represented (three agencies are considered, namely Social Security, Social Insurance, and Chambers of Commerce). The two main interventions are shown in the right hand side of Figure 1:

1. The common front office layer has the goal of reengineering the relationship between users and administrations, filtering the differences among the administrative procedures and virtually showing the users a unique seamless set of agencies. The role of the interface is also to provide a multichannel access and specialized interface for similar group of users, allowing different users preferences and skills.
2. The common back office layer allows public administration to interact through a cooperative infrastructure, in order to exchange data and knowledge already available on businesses, in such a way as to reach two different objectives, i.e. drastically reduce the burden for the business and allow public administration to achieve a common and shared representation of the businesses. On top of this layer, an *event notification* infrastructure is placed, in which the goal is to guarantee synchronization between administrative update events.

In the G4B project, a central role is played in the above architecture by the repository of services. In order to populate the repository with real services to businesses, we have analyzed on the one hand the available documentation, on the web site of the Italian Department for innovation and technologies, that centrally coordinates e-Government projects, and, on the other hand, the many e-Government projects developed by local public administrations. The outcome of this activity is a registry of 450 government to business services classified in terms of about 20 descriptive properties (a snapshot of the registry is provided in Figure 2). Examples of properties are *service description*, that provides a short natural language functional description of the service, and *european cluster*, a classification of services according to the European guidelines proposed in [1]. The registry has been the basis for the design of the repository, that provides a much richer characterization of services and related business processes (see Section 3) and, at the top of the repository, an ontology of services (see Section 4).

| Service Description  | Service Provider        | Type of Users | European Cluster | # of Potential Users (National) | # of Potential Users (Local) | Yearly Service Frequency | Level of automation (actual) |
|--|-------------------------|---------------|------------------|---------------------------------|------------------------------|--------------------------|------------------------------|
| Fundings for agriculture   | Province of Lecce       | Businesses    | Return           | 5.000.000                       | 1.400.000                    | 6.000.000                | 25%                          |
| Communication of the end of an activity to Chambers Of Commerce and Social Insurance | Municipality of Livorno | Businesses    | Registration     | 5.000.000                       | 2.050.000                    | 420.000                  | 50%                          |

**Fig. 2.** An example from the registry of services to businesses provided by italian Public Administration

### 3 Role of a Repository of Services in the New Architecture

In order to be effective, the architecture discussed in Section 2 has to be provided with several knowledge bases and related services. The main goal of the repository (repository G4B in the following) is to describe the services provided from agencies to businesses in such a way to allow its effective use in the two layers of the architecture shown in Figure 1, namely:

1. the *front office layer*, in order to: (i) simplify the service identification by hiding details on the administrations involved in service delivery, and (ii) allow businesses to choose “effective” services from the point of view of their business processes and goals;
2. the *back office layer*, to allow how to plan the re-engineering of administrative processes and make more efficient the development of new services.

The repository G4B has been designed in such a way that it can be useful in principle for three types of players, namely public administrations, businesses, and service developers and providers. To introduce this issue, see in Figure 3 an high level representation of the conceptual schema of the repository G4B, and the three parts of the schema being of specific interest for public administration, businesses and providers. Services are represented both at an abstract level (entity *Service*), and at a concrete level (entity *Concrete Service*) that corresponds to real services offered by providers. On the left hand side the businesses and the internal processes that drive the request of a service supplied by a public administration are represented. To the best of our knowledge this is an innovative characteristics of a repository: services are related with business processes that need them. The relationship between businesses from one side, and agencies and

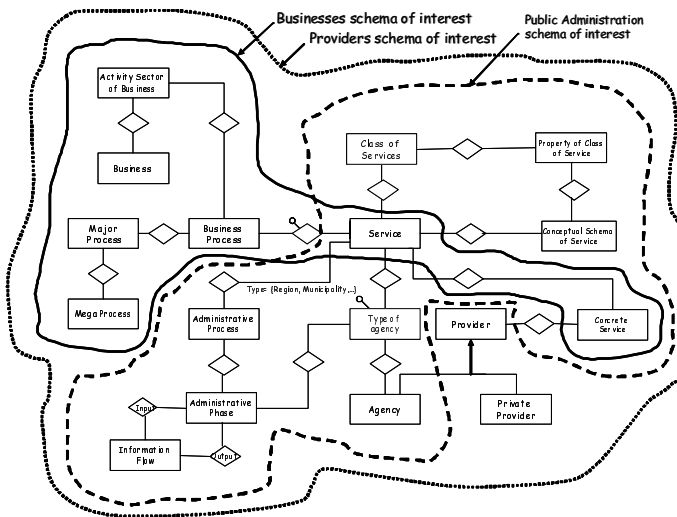


Fig. 3. Conceptual schema of the repository, with the parts of interest for the different players

providers from the other side is seen as the means for businesses to improve efficiency and effectiveness of their business processes.

In order to share a common knowledge, the supplier (agencies and private providers) and the requester (businesses) must refer to a shared representation of the service, based on the service ontology shown on the top, right hand side of Figure 3. In the following section we detail the description of the ontology of services, that provides a common representation of services and their properties on the basis of a common classification scheme.

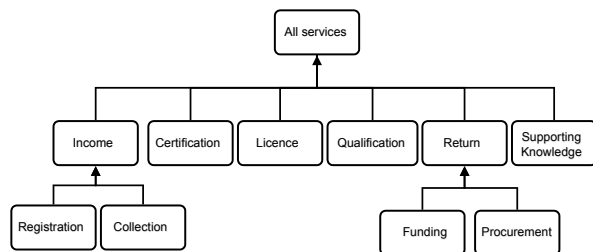
## 4 The Ontology of Services

Due to the issues raised from the needs of knowledge sharing, integration and reuse, in recent years the interest on ontologies is grown not only in the academic field but also within industry and government. There are two most quoted definitions of ontology, namely:

1. the Gruber [2] definition as “*a formal, explicit specification of a shared conceptualization*”;
2. the refinement from Guarino [3] who pointed out the difference between: (i) an ontology as a logical theory accounting for the intended meaning of a formal vocabulary and (ii) an ontology as an *engineering artifact*, constituted by a specific vocabulary and by the assumption on the intended meaning of the vocabulary terms [3].

Other proposals worth mentioning appear in [4] and [5]. In [6] the difference between ontologies and other models such as the Entity Relationship model [7] are discussed, pointing out to the overlapping in expressivity between them. In conclusion, the different approaches to ontologies cover a spectrum that goes from *lightweight* ones, consisting of a list of terms, to formalized logical theories. The core of an ontology is often represented by means of a taxonomy or lattice of classes and subclasses of the concepts needed in order to describe the domain of interest by means of a set of properties referring to relationships, attributes and the related constraints between them.

In our proposal, we define as ontology the model underlying the repository, referring to the lightweight side of the spectrum proposed in [4] and in [5]. We first represents a taxonomy of classes of services with their related properties. The starting point, indeed, is a classification of italian e-Government services to businesses. A second type of



**Fig. 4.** Classes of the ontology of services

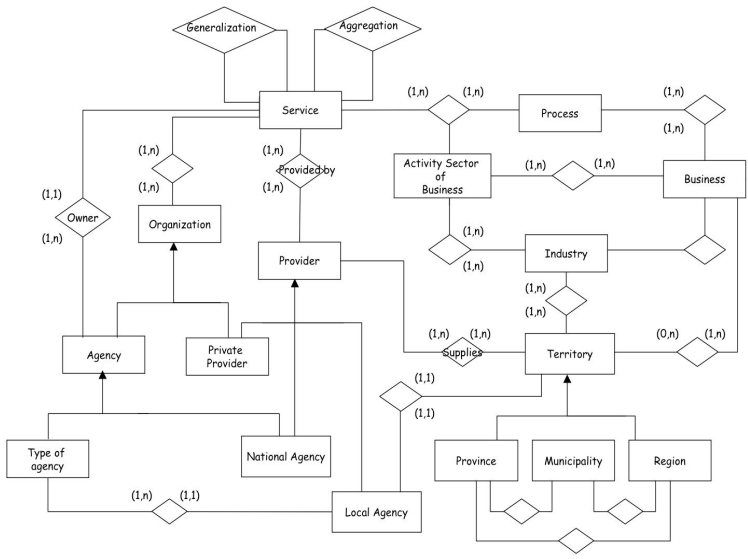


Fig. 5. Conceptual schema of services and processes

classification is introduced whose goal is to define *classes* of services on the basis of their functional characteristics (e.g. a service that produces as output a certificate is an instance of the class *certification*). An analysis of these classes of services revealed the existence of subset relationships among them, that corresponds to the taxonomy of service classes as shown in Figure 4. The taxonomy of classes with their related properties, represents the meta-level of our ontology of services, whose concepts and relationships are represented in the conceptual schema shown in Figure 5.

In functional terms, each class of services is described by three types of properties: (i) *requirements*, the events that activate the service provision, (ii) *preconditions*, the conditions needed to activate the service, and (iii) *effects*, the outcome of the service. A definition domain corresponds to each property, which refers to a set of concepts

| ClassName   | Role          | Requirements         |
|-------------|---------------|----------------------|
| Procurement | Actor         | Owner                |
|             | Process       | SuppliedProcess      |
|             | Resource      | AdmittanceLevel      |
|             | Resource      | Tool                 |
|             | Resource      | Matter               |
|             | Actor         | Beneficiary          |
|             | Resource      | Reason               |
|             | Resource      | OrganizationalUpdate |
|             | Actor         | Provider             |
|             | Preconditions |                      |
|             | Actor         | Applicant            |
|             | Actor         | Provider             |
|             | Actor         | Producer             |
|             | Constraint    | RequestDate          |
|             | Constraint    | ExpiryDate           |
|             | Actor         | Beneficiary          |
|             | Constraint    | RequestedSum         |
|             | Effects       |                      |
|             | Resource      | Outcome              |
|             | Resource      | AllocationDate       |
|             | Constraint    | Duration             |
|             | Resource      | Contents             |
|             | Constraint    | AllocatedSum         |
|             | Constraint    | BeginningDate        |

Fig. 6. Description of the properties of the Procurement class

retrieved from the service conceptual schema describing the service domain. This schema represents the concepts, and the relationships between concepts involved in the service provision. In our context, such a schema is described using the Entity Relationship model. Moreover, each property value has a meta description that defines the *role* played in the service conceptual schema, the role can be *Actor*, *Process*, *Resource*, and *Constraint*. We now provide a description of properties for the *Procurement* class of services (see Figure 6).

A service in the class *Procurement* has: (i) a Requirement property with value *Owner*, whose Role is *Actor*, (ii) a Precondition property with value *RequestDate*, whose Role is *Constraint*, (iii) an Effects property with value *Outcome*, whose Role is *Resource*. This is a very high level description of services, while is out of the scope of our proposal to design a core ontology of services as described in [8].

## 5 Usefulness of the Repository for Providers

Service providers are a first type of player that may take profit of the knowledge represented in the repository G4B. Providers may decide to operate in two major business areas: (i) development of services, and (ii) development of frameworks for integration of services provided by third parties, namely agencies or other providers.

The repository and the related ontology represent different types of knowledge that may be useful for such a goal. Besides properties mentioned in Section 2, other relevant properties represented in the repository concern (see again Figure 2):

- the *provider(s) of the service*, that is the agencies and the private organizations that provide the service to businesses;
- the *type of users*, the classes of potential users interested in requesting a service (e.g. farmer in Figure 2);
- the *number of potential users*, the size of the universe of users interested in requesting the service (both at national and local level);
- *service frequency*, that estimates the number of service requests in a year;
- *level of automation*, that measures the level of implementation of the service. This last property allows to compare different implementations of a service in order to evaluate the residual work to undertake for completing the development process.

Such properties may be used by providers in different strategic decisions, for instance:

- support business predictions in terms of (i) the size and characteristics of the potential market and (ii) the attitude of the users to spend money for a service;
- planning most effective development processes of new services on the basis of the present level of development of services in the different administrative and territorial areas.

## 6 Usefulness of the Repository for Public Administrations

In the context of public administration, building a repository of services aims at creating value for public administration in terms of potential reuse of the services produced and business process reengineering opportunities.

Concerning reuse, in many countries local administrations are organized in terms of different administrative layers (such as municipalities, regions, districts, etc.). In force of laws that discipline the competencies of the administrations, the different administrations have to provide similar services. Since the cost of development and of technological infrastructure covers typically 50-70 % of the total cost of development, significant savings can be obtained coordinating the service development, by (i) developing each service once for all, and (ii) developing first abstract patterns for the classes of services, and then specializing such abstract patterns for specific services in the class.

A second kind of use of the repository of services refers to business process reengineering. Here the goal is to reorganize services using the technological infrastructure described in Figure 1. The presence in the classes of the ontology of similar services among different administrations is an indication that such set of services can be re-organized using cooperation among administrations. For example, the start up of a business and all variations in relevant data, such as address, type of activity, have to be communicated to Chambers of Commerce, Social Security and Social Insurance. In the traditional, non-integrated setting, the burden of business transactions is shared between businesses and the PA. Businesses are estimated to spend about 180 million euros a year just to notify various agencies of their inception and variation events. This cumulative cost corresponds to communicating business identification data relative to 2 million events, using about 3 person/hour for each event (see [9] for the details). On the public administration side, the cost to a single agency for handling the inefficiency is no less than 10 million/year euros. Assuming that each business's data appear in the data bases of at least 10 agencies, this brings the total cost to 100 million euros or more. A further consequence of the autonomy and heterogeneity among the agencies is an estimated 20 to 25 percent of misalignment between business records maintained in multiple registries. This misalignment brings about two additional costs. First, agencies must spend an estimated 20 million euros to reconcile records using clerical review, e.g. to manually trace businesses that cannot be correctly and unequivocally identified within the PA. Second, because most tax fraud prevention techniques rely on cross-referencing records over different agencies, misalignment may result in undetected tax fraud (of currently unknown proportions). In order to enable agencies to offer this service through the common infrastructure shown in Section 2, a business processes reengineering can be undertaken. Specific agencies can be selected as front-end entry points to businesses for specific types of services. In our example, Chambers of Commerce can be involved in updates related to administrative information; when receiving as the data owner an update information from a business, Chambers of Commerce publish it in the event notification infrastructure shown in Figure 1; the information can be subscribed by all other agencies that are interested in the update, such as Social Security and Social Insurance, thus ensuring the correctness, consistency and currency of the business data in the different registries of the agencies. The reengineering of these types of services results in a cost benefit balance extremely fruitful for public administrations.

## 7 Usefulness of the Repository for Businesses

Usually public administrations in their e-Government projects tend to develop *administrative services*, strictly related to their administrative procedures and laws to be

enforced, disregarding the importance of *value added services*, that provide added value to business processes. An example of administrative service is the provision of a certification, while an example of value added service is a statistics or else territorial information that may be useful for marketing processes. In the repository both types of services, must be included. In order to create utility for businesses, services have to be related to business processes using them. In the repository G4B we provide a representation of business processes based on the Porter value chain. In [10] Porter addresses the interplay of cost and differentiation, as two major types of competitive advantage, with the scope of businesses's activities. On that basis, Porter introduces in [10] the *value chain* as a tool for diagnosing and enhancing the competitive advantage of a business by its disaggregation into its most relevant activities. *Value activities* at an high level are classified in two types, the *primary activities*, that are involved in production, sales and assistance and *support activities*, that support the former. Taking that into account, an important issue in Porter's model is related to the relationships or *linkages* [10] between cost and performances of the different value activities. Referring to the conceptual organization of the repository (see again Figure 3), in order to enable the above approach, we have to introduce descriptive attributes of services and processes that support predictions in terms of usefulness of a service for a process. The values of such attributes for specific services and processes are obtained by analyzing two different knowledge sources: (i) the business specifications, in terms of the above mentioned sources, resulting from theoretical and empirical analyses performed on specific areas of businesses by Chambers of Commerce and other domain experts; (b) if made available, the opinions provided by businesses who previously experimented the usefulness and effectiveness of services described in the repository. Weights are defined to set a decision procedure for choosing the maximal value suite of services in a given business process.

As a consequence of the above discussion, the repository, on the one hand, provides knowledge to businesses related to administrative services, namely prescriptive public administration obligations involved in processes; on the other hand, helps businesses in the discovery of value added services used in processes, not explicit related to norms. In their interaction with agencies, businesses, starting from a business process flow, need to discover all interaction events related to the process flow, both referring to administrative services and to value added services, in order to plan the choice of the best suite of services provided by administrations, also on the basis of the cost/effectiveness ratio. As an example, we refer to the *Opening new sales points* process as a subprocess of the "Business Infrastructure-strategic planning-territorial development" chain. An example is given in Figure 7 where the three subprocesses of the *Opening new sales points* process are shown. We consider the case of a business operating in the furniture industry. Focusing on the *Location Research and Choice* subprocess, we consider the *Data Collect* activity, whose goal is to collect information in order to choose the most competitive location for a new point of sales. For this goal, the *Data Collect* activity has to retrieve information about the territory (e.g. number of inhabitants, number of resident households, number of supermarkets), and about competitors (e.g. number of points of sales in the same territory, commercial data about their selling, etc.). This knowledge is available in the data bases of the local agencies and of the local Chambers of Commerce. In this context, the repository acts as a mediator, associating the service



“retrieve information on territorial marketing” to the *Data Collect* activity. Referring to the architecture described in Sections 2 and 3, the requester, namely the furniture business, can access services needed through the common front office layer (see Figure 1), where service description is enriched with the link to the associated process in the value chain. Once identified the service, the corresponding query is issued through the back office layer (see again Figure 7). In such a way both service identification and service provision are made more effective.

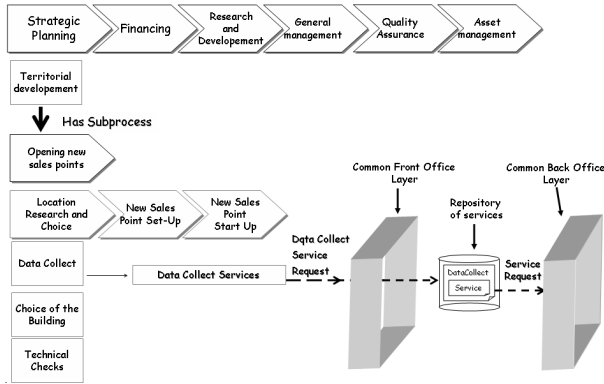


Fig. 7. Example of use of the process-to-service relationship

## 8 Related Work

A proposal for repository technology appears in [11]. In the database area, the conception of repository as dictionary has a tradition in model management (see [12]). Integration of services, schemas, software components, etc. is a key issue in all information systems where several levels of cooperation have to be established between different organizations or players [13]. Recently integration is investigated in the area of ontologies and web services. The lack of semantics in registry of services such as the web services standard UDDI [14] enhances the development of tools that make more effective service composition (see [15]). A number of issues related to semantic integration is discussed, and techniques for integrating disparate resources are provided making use of ontologies, see [16]. Repositories of conceptual schemas are proposed in several application areas; e.g. in biosciences [17], for reuse in schema design in [18]. A data repository is used in [19] as the core structure of a mediator-like module. A repository of relational schemas is proposed in [20] within a GLAV data integration system. [21] introduces the concept of conceptual schema package as an abstraction mechanism in the ER model. Repositories of ontologies are proposed in several papers, in the area of Semantic Web [22] and in the Web Services area [8]. The alignment and integration of ontologies is investigated e.g. in [23], where information integration is enabled by having a precisely defined common terminology. The literature on the application of ICT technologies in e-Government is vast. In [24] a conceptual framework is proposed, identifying the importance, categorisation and presentation of the strategies for overcoming

technical and organizational challenges facing a transactional e-Government system. With regard to the technological architecture, [25] compare architectures that can be adopted in Government applications, and [26] propose basic architectural elements of a forecasting system for e-Government. The role of a service repository in platforms for one-stop Government is discussed in [27]. The issue of value creation in industrial business strategy has been recently investigated for business process reengineering in the public administration. Referring to the Porter value chain [10], case studies are discussed in [28] and in [29].

## 9 Future Work

CM Group is now implementing the repository and the related ontology whose general organization has been discussed in this paper. Several functionalities have to be refined, that enable the different usages discussed, such as reuse, business processes reengineering, identification of value added services, identification of services associated to a given business process, planning of the most effective development process. In the future we aim at focusing in the area of heuristics and tools for efficient production of the service conceptual schemas, as introduced in Section 4. To conceive such heuristics we will adapt methodologies and tools presented in [30].

## References

- [1] Italian Government: The Italian Initiative on E-Government for Development. The Reference Model: E-Model. (2002)
- [2] Gruber, T., R.: A translation approach to portable ontology specification. *Knowledge Acquisition* (5) (1993)
- [3] Guarino, N., ed.: Formal ontologies and information systems. In Guarino, N., ed.: *Proceedings of FOIS'98*, IOS Press (Amsterdam, 1998)
- [4] Uschold, M.: Knowledge level modelling: concepts and terminology. *Knowledge Engineering Review* (1998)
- [5] Uschold, M., Gruninger, M.: Ontologies: Principles, methods and applications. *Knowledge Engineering Review* **11**(2) (1996) 93–155
- [6] Uschold, M., Grüninger, M.: Ontologies and semantics for seamless connectivity. *SIGMOD Record* **33**(4) (2004) 58–64
- [7] Elmasri, R., Navathe, S.: *Foundamentals of database systems*, Fifth Edition. Addison-Wesley Publishing Company (1994)
- [8] Mika, P., Oberle, D., Gangemi, A., Sabou, M.: Foundations for service ontologies: aligning owl-s to dolce. In: WWW. (2004) 563–572
- [9] Bertolletti, M., Missier, P., Scannapieco, M., Aimetti, P., Batini, C.: Improving Government-to-Business Relationships through Data Reconciliation and Process Re-engineering. In Wang, R., ed.: *Information Quality - Advances in Management Information Systems-Information Quality Monograph (AMIS-IQ) Monograph*. Sharpe, M.E. (2005). Shorter version also in ICIQ 2002.)
- [10] Porter, M.E.: *Competitive Advantage*. The Free Press, New York (1985)
- [11] Bernstein, P.A., Dayal, U.: An overview of repository technology. In: VLDB '94: *Proceedings of the 20th International Conference on Very Large Data Bases*, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (1994) 705–713

- [12] Atzeni, P., Cappellari, P., Bernstein, P.: A multilevel dictionary for model management. *Lecture Notes in Computer Science* **3716** (2005) 160–175
- [13] Themistocleus, M., Chen, H.: Investigating the integration of smes' information systems: an exploratory case study. *International Journal of Information Technology and Management* **3**(2/3/4) (2004) 208–234
- [14] UDDI.org: UDDI Technical White Paper. Available on line (link checked October, 1st 2001): [http://www.uddi.org/pubs/lruUDDI\\_Technical\\_Paper.pdf](http://www.uddi.org/pubs/lruUDDI_Technical_Paper.pdf) (2001)
- [15] Hull, R., Su, J.: Tools for composite Web Services: a short overview. *SIGMOD Record* **34**(2) (2005)
- [16] McGuinness, D., L.: Ontological issues for knowledge-enhanced search. In: *Frontiers in Artificial Intelligence and Applications*,. IOS Press, Washington, DC (1998)
- [17] Taxonomic Databases Working Group Annual Meeting: Taxonomic Databases Working Group on Biodiversity Informatics, University of Canterbury, Christchurch, New Zealand, Taxonomic Databases Working Group Annual Meeting (2004)
- [18] Ruggia, R., Ambrosio, A.P.: A toolkit for reuse in conceptual modelling. In: *CAiSE*. (1997) 173–186
- [19] Palopoli, L., Terracina, G., Ursino, D.: Dike: a system supporting the semi automatic construction of cooperative information systems from heterogeneous databases. *Software Practice and Experience* **33**(9) (2003) 847–884
- [20] Motro, A., Anokhin, P.: Fusionplex: resolution of data inconsistencies in the integration of heterogeneous information systems. *Information fusion* (2004)
- [21] Shoval, P., Danoch, R., Balaban, M.: Hierarchical entity-relationship diagrams: the model, method of creation and experimental evaluation. *Requir. Eng.* **9**(4) (2004) 217–228
- [22] Oberle, D., Staab, S., Studer, R., Volz, R.: Supporting application development in the Semantic Web. *ACM Transactions on Internet Technology* **5**(2) (2005) 328–358
- [23] Wang, J., Gasser, L.: Mutual online concept learning for multiple agents. In: *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, New York, NY, USA, ACM Press (2002) 362–369
- [24] Madi Al-Sebie, Zahir Irani: Technical and organisational challenges facing transactional e-government systems: an empirical study. *Electronic Government, an International Journal* **2**(3) (2005) 247–276
- [25] Batini, C., Mecella, M.: Enabling Italian e-Government Through a Cooperative Architecture. *IEEE Computer* **34**(2) (2001)
- [26] Nikolopoulos K., Patrikakis C.Z., L.B.: Forecasting systems for e-government. *Electronic Government, an International Journal* **1**(4) (2004) 374 – 383
- [27] Wimmer, M.A.: A european perspective towards online one-stop government: The egov project. *Electronic Commerce Research and Applications*, **1** (2002)
- [28] Hagel, III., J., Brown, J., S.: Your next IT strategy. *Harvard Business Review* (2001) 106–113
- [29] Andersen, K., V.: Reengineering public sector organizations using information technology. *Reinventing Government in the Information Age* (1999)
- [30] Batini, C., Garasi, M.F., Grosso, R.: Reuse of a repository of conceptual schemas in a large scale project. In: *Advanced Topics in Database Research*, Idea Book (2005)

# Towards Automatic Integration of Persistency Requirements in Enterprise-Systems – The Persistent-to-Persistent Patterns

Mira Balaban<sup>1</sup> and Lior Limonad<sup>2,\*</sup>

<sup>1</sup> Computer Science Department

<sup>2</sup> Information Systems Engineering Department  
Ben-Gurion University of the Negev, Beer-Sheva 84105, ISRAEL  
mira@cs.bgu.ac.il, limond@bgu.ac.il

**Abstract.** Nowadays, implementing the *Business-Data* layers interaction is assisted by industry tools that provide abstraction on top of concrete database systems, but still requires writing a large amount of annoying bug-infected code. Full automation of the *Business-Data* layers interaction will provide a great improvement in complex systems development.

We claim that automatic integration of the *Business-Data* layers requires careful analysis of navigational structures that involve persistency concerns in the *Business* layer. Therefore, our approach consists of a set of independent *Data Access Patterns*, each applying to a specific navigational structure. Moreover, the patterns are *Business* layer transparent, i.e., the *Data* layer interaction leaves the *Business* layer intact.

In this paper we introduce two *Data Access Patterns* termed *Persistent-to-Persistent* patterns that handle navigational structures between classes that are both marked as persistent. The patterns are notable for their handling of *Persistent-to-Persistent* interaction, where all data is persistently stored. All patterns are based on a core *Proxy Data-Mapper* pattern, that is shortly described.

**Keywords:** Persistency, Design Patterns, Transformation Framework, Model-Driven Approach, UML, Models Co-evolution, Data Source Layer, Domain Layer, Composite Transformations, Refactoring.

## 1 Introduction

Modern enterprise systems are layered, having at least *Presentation*, *Business* and *Data* layers, that coexist, co-evolve, and interact during a system's life cycle. While most layer separation reflect separation of responsibilities, the *Business-Data* separation reflects storage concerns, implying complex interaction between the layers, due to information duplication and overlapping responsibilities. Moreover, the combined operation of the *Business* and the *Data* modules usually

---

\* Supported by the Lynn and William Frankel center for Computer Sciences.

suffers from model mismatch problems, and raises problems of consistency. Considering today's Data layer technology, it is impractical to assume that both layers will adhere to the same technical space.

Nowadays, implementing the *Business-Data* interaction is assisted by industry tools that provide abstraction on top of concrete database systems, but still requires writing a large amount of annoying bug-infected code. Full automation of the *Business-Data* layers interaction will provide a great improvement in complex systems development, and goes along with the rising *Model-Driven Architecture* (MDA) approach [1] that strives towards automatic development and integration of software. Existing technologies offer partial solutions for persistency insertion. Tools like Sun's JDO (Java Data Objects) [2, 3], J2EE [4, 5] and Java's Hibernate use embedded SQL and pre-compiled meta markings (usually using XML files) for managing persistent entities. These tools let the developer design the structure of the persistent layer. The latter is usually based on widely discussed two-way *object-relational* transformations for mapping between the models (consult [6, 7, 8, 9, 10, 11, 12, 13]).

We claim that automatic integration of the *Business-Data* layers requires careful analysis of navigational structures that involve persistency concerns in the *Business* layer. Overall automation of these layers interaction can be obtained by repeated automation of the *Data* interaction for individual navigational structure. Therefore, our approach consists of a set of independent *Data Access Patterns*, each applying to a specific navigational structure. The automatic *Business-Data* layers integration will have the following steps:

1. **Persistency marking:** Business layer objects that should be persistent are marked (e.g., stereotyping on the associated class diagram model).
2. **Data layer specification:** Persistent layer structure is specified, based on existing mappings between domain to persistent models, and using currently available tools.
3. **Data access insertion:** An intermediate *data access layer* that integrates the business layer with the persistent layer is introduced, by applying the data access patterns.
4. **Developer review:** The developer can review and possibly revise the new code.

The ideal integration should leave the business layer intact. This requirement is justified on a conceptual viewpoint, and essential for persistency automation, since it means that changes involve only the new classes on the data access layer. Otherwise, business classes, whose structure is unknown, should be modified, which is difficult to automate. Indeed, the patterns that we suggest are all *Business* layer transparent, i.e., the *Data* layer interaction leaves the *Business* layer intact.

Our work can be viewed as a continuation and further development of Fowler's work [14]. Fowler suggests two patterns for the integration of a persistent layer with a domain layer: The *active record* and the *data-mapper* patterns. In the first pattern, responsibility for persistency services is imposed on the persistent class itself, yielding a business class that is extended with persistency services.

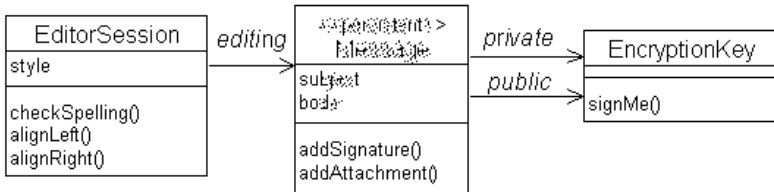
The data-mapper pattern suggests to insert a mediator data-mapper class between a persistently marked class  $P$  and the actual persistent storage. This pattern reduces the persistency responsibilities imposed on  $P$ , but still requires the addition of an object ID attribute to  $P$ . Fowler's patterns do not refer to context classes of  $P$ , i.e., classes whose instances have reference to, or are referenced by instances of  $P$ . Automation of the *Business-Data* layers interaction requires handling ingoing and outgoing references, into and from, a persistent instance.

In this paper we introduce two *Data Access Patterns* termed *Persistent-to-Persistent* patterns that handle navigational structures between classes that are both marked as persistent. The patterns are notable for their handling of *Persistent-to-Persistent* interaction, where all data is persistently stored, while *Business* layer classes are still intact. The solutions suggested by the patterns can be applied at any stage of system development. All patterns are based on a core *Proxy Data-Mapper* pattern, that is shortly described.

Section 2 describes the core *Proxy Data-Mapper* pattern, that handles a single persistent class in a context independent way. In section 3 we describe the two *Persistent-to-Persistent* patterns, that handle persistent context of a persistent class. Section 4 is the conclusion and includes discussion of future research.

## 2 The Proxy Data Mapper Pattern

The *Proxy Data Mapper* pattern can be viewed as an elaboration of Fowler's *Data Mapper* pattern, in a way that leaves the *Business* layer classes intact. For example, Consider an email client application, as illustrated in Figure 1. Applying Fowler's *Data Mapper* pattern to this case is described in Figure 2. In



**Fig. 1.** A persistency marking situation: A persistent class with in-memory clients and outgoing references

the resulting design, instances of the persistently marked class are given object IDs, the client class *EditorSession* is now aware of the new *MessageDataMapper* class, and no solution is provided for outgoing references of *Message* instances. Therefore, this pattern provides only a partial solution for achieving automation of the *Business-Data* layers integration.

Following GoF spirit [16], we suggest a solution, called *Proxy Data Mapper*, as in Figure 3. Its main idea is hiding the existence of the concrete mapper class

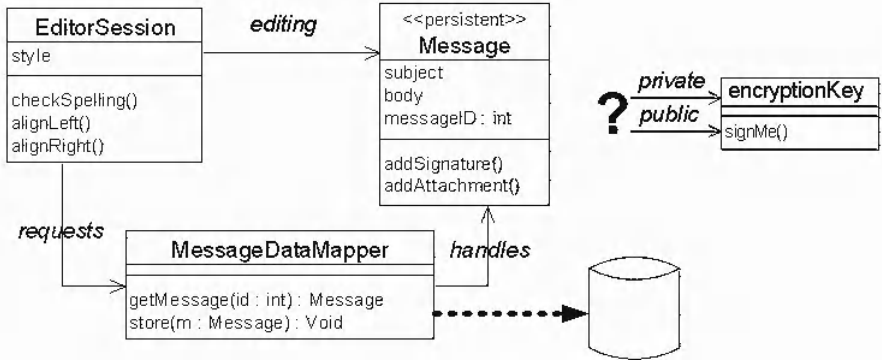


Fig. 2. Applying Fowler’s data mapper pattern

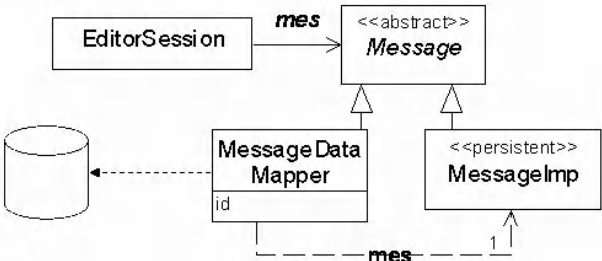


Fig. 3. Combining the Mapper and Proxy concepts together

from the client, while still having a mapper class that wraps the persistent class with persistency services. *Proxy Data Mapper* combines the mapper pattern of [14] with the *proxy* pattern of [16].

The *Proxy Data Mapper* pattern renames the persistent class *Message* to be *MessageImp*. It extracts the interface of the persistent class and defines it as a new interface/abstract class with the original name (*Message*). *MessageDataMapper* and *MessageImp*, both implement this interface. Clients of *Message* (e.g., *EditorSession*) communicate with the concrete message (*MessageImp* in the figure) through the interface. Therefore clients are not affected. The "trick" here is that the real object that clients communicate with is an instance of *MessageDataMapper*, while their type signature knows only about *Message*.

The *MessageDataMapper* class encapsulates both a concrete message (now termed *MessageImp*), and its identification in the database. Synchronization between in-memory and persistent objects is imposed on the new *MessageDataMapper*. A request to the mapper to execute a method on its encapsulated persistent object triggers three actions: (1) **load** the persistent object from storage to memory; (2) **forward** the requested operation to the loaded object; (3) **update** the state of the loaded object in the storage. In a simple implementation,

the loaded persistent object is instantiated only for the execution of the delegated method. After the execution terminates, the instance is released from being memory resident. This temporary existence of the persistent object is expressed as a dashed line arrow that connects the mapper class and the persistent class in figure 3. A smarter *Datamapper* implementation can save redundant database interaction by caching loaded persistent objects.

The code generated by applying *Proxy Data Mapper* involves the construction of the new *Data Access* layer classes alone, since *Business* layer classes are not changed. However, prior replacement of the persistent class constructors by corresponding factory methods is necessary because creation of instances of the persistent class should be replaced by creation of their in-memory data-mapper representative instances. Clients of the persistent class will apply the factory method, and will not be aware that the instance they are holding is actually a data mapper instance.

The construction of the implementation class constructs a subclass of the persistent class, and moves all structure and methods of the persistent class into the new implementation class (apart from the factory methods). The heaviest part of this refactoring is carried by the construction of the datamapper subclass, which handles both storage services (load, store, update, delete), and delegation to the implementation class tasks. Note that since the thin in-memory datamapper object represents the stored object, destruction of the datamapper object means deletion of the stored object (the datamapper is the only link to the stored object). Actual implementation of destruction is language dependent (in C++ it is done via the object destructor; in Java it is done via the finalize service).

The top level description of the *Proxy Data Mapper* code transformation is given below. A detailed description appears in [17].

```
ProxyDataMapper(Class persistentClass, DB dbSchemaElement){
// 1. Apply factory method pattern:
    replaceConstructorsWithFactoryMethods(persistentClass);
// 2. Extract the implementation from the persistentClass:
    persistentClassImp = extractPersistentClassImp(persistentClass);
// 3. Construct a data mapper class that connects the persistent object with
//    its memory mirror:
    constructPersistentClassDataMapper(persistentClassImp, dbSchemaElement);
// 4. Tag modification:
    stereotype(persistentClass, 'abstract');
    removeStereotype(persistentClass, 'persistent');
```

### 3 Persistent-to-Persistent Patterns

The *Proxy-Data-Mapper* pattern is used for wrapping all persistently marked classes with persistency services. The comprehensive solution to persistency requirements is achieved by applying a set of *Context Data Access Patterns*. The *context* singles out the persistent class context in the *business* layer to which the



pattern applies. The *context* of the persistently marked class is determined by a combination of the following three parameters:

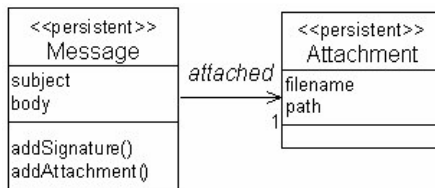
1. **Navigation:** The direction of the association which connects the persistent class and the context class.
2. **Cardinality:** This parameter makes a distinction between an association to a single object (marked with "1") and an association to multiple objects (marked with "\*").
3. **Context class:** The context class may be stereotyped as being either an in-memory class or a persistent one.

The combinations of these three parameters give rise to six different *context data access patterns*.

In this paper we focus on two patterns that correspond to the *Persistent-to-Persistent* context in which the context class is persistent. These patterns are particularly interesting since it is tempting to think that both classes being persistent, it is sufficient to apply some persistency automation tool (e.g., a *object-relational* mapping tool), that provides persistent account for both classes and the association between them. However, such database integration tools do not provide application level support for the stored objects so that the *Business layer* is oblivious to persistency considerations.

### 3.1 Persistent to Persistent

This pattern applies to cases where a single instance of a persistent class references a single instance of another persistent class, using a permanent reference, i.e., using a reference attribute. For example, in the e-mail client application example, each *Message* object may contain an attached file that is expressed as an instance of the class *Attachment* as illustrated in Figure 4.



**Fig. 4.** Persistent to Persistent - Context: Message is the persistent class, and Attachment is the context class

Since the persistency situation of both classes is symmetric we can think of either class as *the persistent* class, while the other one plays the role of the context class. Since the major load of solving the problems seem to fall on the class that holds the reference, we refer to the latter as *the persistent* class, and to the referenced class as *the context* class. In Figure 4, *Message* is *the persistent* class, while *Attachment* is *the context* class. Clearly, in a bi-directional reference situation a

class can take both roles. Also, the same context class instance may be shared by multiple references that exist either in-memory or in-storage. The context of the *Persistent to Persistent* pattern is identified by the following combination:

| Navigation | Cardinality | Context class |
|------------|-------------|---------------|
| outgoing   | 1           | persistent    |

### Problems that arise in the context of the Persistent to Persistent pattern

The context of the *Persistent to Persistent* pattern leads to problems that are caused by the need to manage duplicated in-memory and in-storage references. They arise when a persistent object, e.g., a message, that keeps a permanent reference to another context persistent object, e.g., an attachment, is loaded into memory, or is updated in memory, or is deleted from memory, or is created (stored).

1. **The load problem:** Loading the persistent object must *realize* also the permanent reference to its context object. The problem is what sort of realization the context loading should take: Should the context object be fully loaded, or should just a thin representative (e.g., a data mapper) be loaded. In any case, since both objects are loaded in either a full or a lightweight format, the in-storage reference is duplicated. For example, an in-memory copy of a message instance is generated when some *Message* service, e.g., `addSignature()`, is invoked. The message instance must be associated with its attachment instance, implying that the outgoing reference must be realized in memory.
2. **Avoiding in-memory object duplication:** The same problem as in the *Persistent Collection Data Mapper* pattern appear here. In-memory object duplication might occur when loading stored objects that share a single stored object. For example, assume the storage state illustrated in Figure 5. It shows two stored messages that are linked to the same attachment. Loading both messages to memory requires realization of their correlated attachment. This procedure might yield in-memory duplication of the shared in-storage attachment instance. Once duplicated in memory, inconsistencies between storage and memory are unavoidable, since the in-memory copies are independent of each other, and their updates cannot be synchronized.

| Messages           |                       |                      |                      |                 |             |
|--------------------|-----------------------|----------------------|----------------------|-----------------|-------------|
| <i>subject</i>     | <i>body</i>           | <i>Attachment_FK</i> | Attachments          |                 |             |
| Important          | Dear Sir, ...         | 2                    | <i>Attachment_PK</i> | <i>Filename</i> | <i>Path</i> |
| Virus Notification | Hi, just wanted to... | 2                    | 1                    | Paper.txt       | C:\myDocs   |
|                    |                       |                      | 2                    | Logo.gif        | D:\graphics |

Fig. 5. Possible storage snapshot

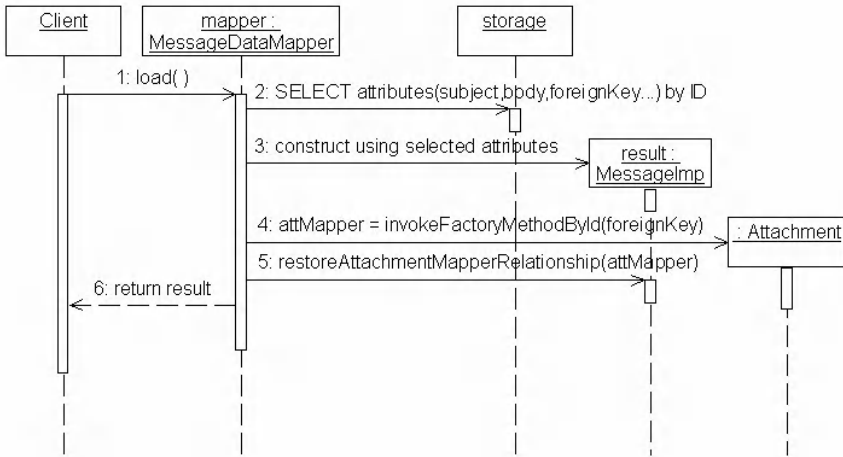
3. **The update problem:** The in-memory copy of the persistent object duplicates the in-storage reference to the context object. Changes made to the in-memory copy and its context must be recorded in storage, including cases of changed context, e.g., when the attachment is replaced.
4. **The deletion problem:** When a full in-memory copy of the persistent object is deleted from memory, e.g., when its services are completed, its in-memory reference to the context object is lost. However, both the persistent and the context objects still reside in storage. If there are no other in-memory references to the context object, it is possible that it is also deleted from memory.

In a simple setting, where all object references reside in memory alone, deletion of an in-memory object which is a copy of a stored object should imply deletion of the real object from storage (as discussed in the context of the Proxy Data Mapper pattern). However, when object references reside also in storage, this is not the case anymore, since it is possible that although all in-memory references to objects are lost, there are still in-storage references. For example, when an in-memory copy of a stored message instance is destructed, while its light weight representative still lives in memory, its reference to the associated attachment instance (which might be a full copy or just a lightweight representative) is lost. This might lead to in-memory destruction of the context attachment instance. However, it should not imply deletion of the in-storage attachment object, since the in-storage reference is still there.

5. **The creation problem:** When the persistent object is created, its context is also created (it is a permanent visibility context). Both objects, being marked as persistent should be stored, and removed from memory, in case they were explicitly created as in-memory objects. The reference must be stored as well; otherwise, it will be lost.

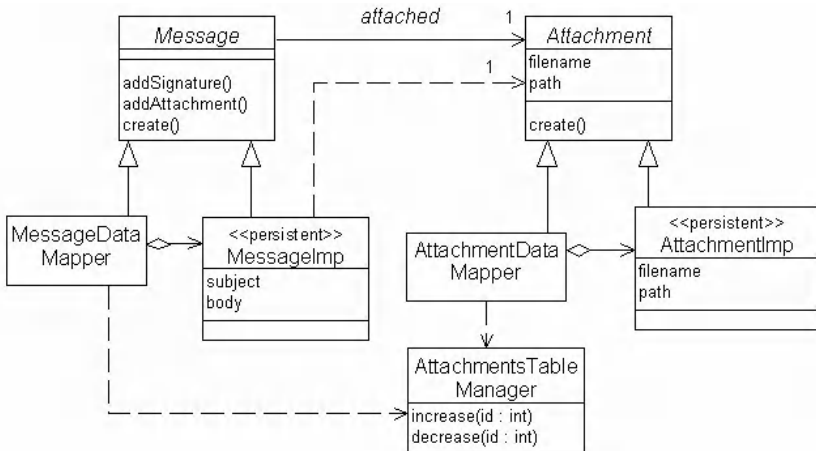
## Problem solutions

1. **Solution to the load problem:** The responsibility to realize the reference to the context object is imposed on the thin in-memory representative of the persistent object (its datamapper). When loading a persistent object to memory, its datamapper also reconstructs a thin representative for the context object and re-attaches it as illustrated in Figure 6. Since both representatives reside in the *Data Access Layer*, the solution does not affect the *Business layer*.
2. **Solution to the object duplication problem:** The same solution described in the *Persistent Collection Data Mapper* pattern is applied here.
3. **Solution to the update problem:** All local changes to the in-memory copies (persistent and context objects) are handled by their corresponding datamappers. Changes in reference to the context object are handled by the datamapper of the persistent object. This is achieved by revising the `store()` and `update()` services of the datamapper of the persistent object, so that they keep track of changes in reference to the context object and update storage accordingly.
4. **Solution to the deletion problem:** To solve this problem, the datamapper of the context class is associated with a new *Data Access Layer* class named



**Fig. 6.** Load implementation of the persistent class mapper

*TableManager*. The *TableManager* class has two responsibilities: (1) tracking reference count to the persistent instance, either from in-memory or from storage. (2) implementing the storage deletion procedure instead of the owning datamapper. A context datamapper that has an associated *TableManager* is revised to delegate the storage deletion procedure to its *TableManager*. In addition, the persistent datamapper should notify the tableManager class about storage reference changes as well. For example, the datamapper of a message should notify the *TableManager* of *AttachmentDataMapper* whenever an attachment is added or removed from its message object. Instances are removed from storage by the *TableManager* only if no in-memory or



**Fig. 7.** Persistent to Persistent - Static Action Result

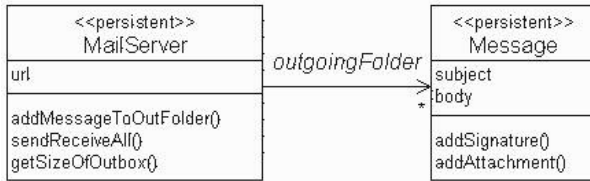
stored references exist. To account for accurate reference tracking, *TableManager* classes are implemented as *singleTons*.

5. **Solution to the creation problem:** Same as for the **update** problem.

Summing up all solutions proposed by the *Persistent to Persistent* pattern, Figure 7 illustrates the static structure of the resulting transformation. Note that indeed, the *Business layer* classes stay intact.

3.2 Multi Persistent to Persistent

This pattern applies to cases where a single instance of a persistent class references **multiple** instances of another persistent class using some collection implementation. For example, in the e-mail client application example, each *MailServer* object may contain a collection of *Message* instances to be sent as illustrated in Figure 8. The context of this pattern is identified by the following combination:



**Fig. 8.** Multi Persistent to Persistent - Context : MailServer is the persistent class, and Message is the context class

| Navigation | Cardinality | Context class |
|------------|-------------|---------------|
| outgoing   | *           | persistent    |

The context of this pattern raises one additional problem, on top of the problems described in the *Persistent to Persistent* context:

**The problem of in-memory realization of a persistent collection:** The multiple cardinality reference must be realized in both layers: persistent and application. The pre-condition of this pattern is that the 1 : *many* cardinality constraint is already implemented in the database, probably using a standard database modeling tool. The problem is to provide an in-memory realization for a 1 : *many* cardinality constraint, where the referenced entities are persistently stored. The collection’s in-memory existence should be reduced to minimum, since its content is already stored. Therefore, only elements that are necessary for collection management should be realized, leaving the real content in storage.

**Problem solution:** A specific collection, e.g., *MessageCollectionMapperImp*, for persistent elements of the context class is introduced. The elements of the new collection do not reside in-memory. The collection is responsible for supporting application level requests for in-memory copies of stored context class

objects. It provides solutions for the problems of **loading**, **deletion**, and **addition** of stored objects. For example, when a full in-memory copy of a *MailServer* stored object requests a specific message, *MessagesCollectionMapperImp* loads an in-memory (thin) copy. When the *MailServer* in-memory copy removes a message from its collection, *MessagesCollectionMapperImp* applies its deletion procedure. The new collection communicates with the table manager of the context class, in order to keep the reference count to stored context elements updated.

Note that the persistent class does not change, as the new collection is hidden being a subtype of a regular library collection class (e.g. *List*). The comprehensive solution for this pattern is illustrated in Figure 9.

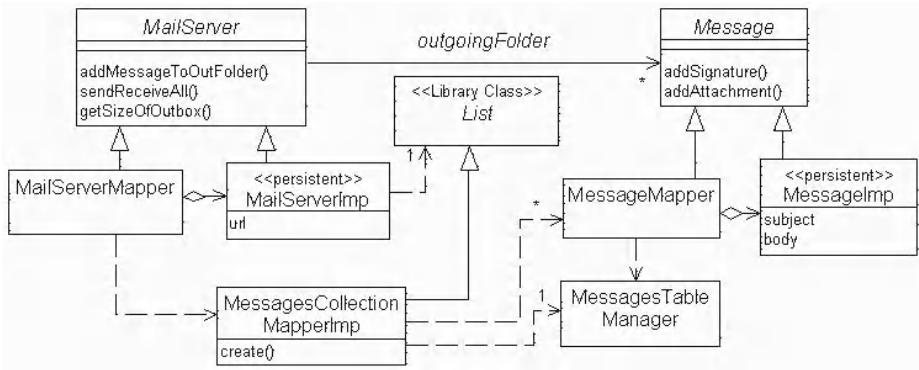


Fig. 9. Multi Persistent to Persistent - Static Action Result

## 4 Discussion

In this paper we have introduced a *pattern-driven* method for the realization of persistency requirements. The *Proxy Data Mapper* core pattern is introduced followed by a pair of patterns corresponding to the Persistent-to-Persistent context. We have started an implementation towards automation of our method. A tool that applies the patterns to a static class diagram is already implemented and in use. We now work on the automation of the code transformation.

## References

- [1] Miller, J., Mukerji, J.: MDA Guide Version 1.0.1. OMG. (2003)
- [2] Ezzio, D.: Using and Understanding Java Data Objects. Apress (2003)
- [3] Russell, C.: JSR 12: JavaTM Data Objects (JDO) Specification. Sun Microsystems, Inc. (2004)
- [4] Hapner, M., Shannon, B.: JSR 151: JavaTM 2 Platform, Enterprise Edition 1.4 (J2EE 1.4) Specification. Sun Microsystems, Inc. (2003)

- [5] Richard, S.: Java Persistence for Relational Databases. Apress (2003)
- [6] Thatten, S.M.: Report on the object-oriented database workshop: Implementation aspects. In: Object-Oriented Programming Systems, Languages and Applications. acm Press (1987)
- [7] Saake, G., Conrad, S., Schmitt, I., Turker, C.: Object oriented database design: What is the difference with relational database design? In: Object-World, Frankfurt (1995)
- [8] Joseph, W., Ralph, E., Quince, D.: Connecting business objects to relational databases. In: Proceedings of the 5th Conference on the Pattern Languages of Programs, Monticello-IL-EUA (1998)
- [9] Ramanatha, C.: Providing object-oriented access to a relational database. Technical report, Mississippi State University, Department of Computer Science (1995)
- [10] Hakimpour, F., Geppert, A.: Resolution of semantic heterogeneity in database schema integration using formal ontologies. Information Technology and Management (2005)
- [11] Blaha, M., Premerlani, W.: Object-Oriented Modeling and Design for Database Applications. Prentice Hall (1998)
- [12] Keller, W.: Mapping object to tables, a pattern language. Technical report, Siemens, Wien, Austria (1998)
- [13] Andersson, M.: Extracting an entity relationship schema from a relational database through reverse engineering. In: Proceedings of the 13th International Conference on the Entity-Relationship Approach. (1994)
- [14] Fowler, M.: Patterns of Enterprise Application Architecture. Addison-Wesley (2003)
- [15] Nock, C.: Data Access Patterns: Database Interactions in Object-Oriented Applications. Addison-Wesley (2003)
- [16] Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns, Elements of Reusable Object-Oriented Software. Addison-Wesley (1994)
- [17] Limonad, L.: Automatic generation of data-access layer. Master's thesis (2006)

# Consistency of UML Class Diagrams with Hierarchy Constraints

Mira Balaban<sup>1</sup> and Azzam Maraee<sup>2,\*</sup>

<sup>1</sup> Computer Science Department

<sup>2</sup> Information Systems Engineering Department  
Ben-Gurion University of the Negev, Beer-Sheva 84105, ISRAEL  
mira@cs.bgu.ac.il, mari@bgu.ac.il

**Abstract.** UML class diagrams are probably the most important, well-established, UML model. They play an essential role in the analysis and design of complex systems. UML class diagrams allow the specification of constraints such as cardinality constraints, class hierarchy constraints and inter-association constraints. Constraints extend the expressivity of class diagrams, but enable the specification of *unsatisfiable* class diagrams, i.e., class diagrams that have no finite non-empty instance world. Nowadays, UML case tools still do not check satisfiability of class diagrams, and implementation languages still do not enforce design level constraints. But the expectation is that in the future, and in particular with the prevalence of the Model Driven Engineering approach, all information in a design model will be effective in its successive models.

In this paper, we present an algorithm for testing the satisfiability of UML class diagrams that include class hierarchies with “disjoint/overlapping” and “complete/incomplete” constraints. The algorithm is based on a reduction to a previous algorithm of Lenzerini and Nobili that was applied only to ER-diagrams without class hierarchies. Our algorithm is simple and feasible since it adds in the worst case only a linear amount of entities to the original diagram. It improves over previous elaboration of the Lenzerini and Nobili method that require the addition of an exponential number of new entities to the original diagram. An implementation of our method within a UML case tool is currently under development.

**Keywords:** UML class diagram, finite satisfiability, cardinality constraints, class hierarchy constraints.

## 1 Introduction

The Unified Modeling Language (UML) is nowadays the industry standard modeling framework, including multiple visual modeling languages, referred to as UML models. They play an essential role in the analysis and design of complex systems. As such, it is important to guarantee that models provide a reliable support for the designed systems. Therefore, an extensive amount of research efforts is devoted to the formal definition of UML models, and the definition

---

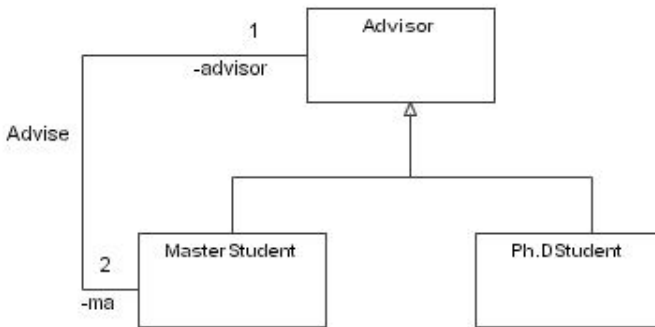
\* Supported by the Lynn and William Frankel center for Computer Sciences.



of correctness in UML diagrams [c.f. [2], [5]] Indeed nowadays, UML case tools still do not test diagram correctness, and implementation languages still do not enforce design level constraints. But the expectation is that in the future, and in particular with the prevalence of the Model Driven Engineering approach, all information in a design model will be effective in its successive models.

Class Diagrams are probably the most important and best understood among all UML models. A Class Diagram provides a static description of system components. It describes systems structure in terms of classes, associations, and constraints imposed on classes and their inter-relationships. Constraints provide an essential means of knowledge engineering, since they extend the expressivity of diagrams. UML supports class diagram constraints such as cardinality constraints, class hierarchy constraints, and inter-association constraints. Example 1 below, presents a class diagram that includes cardinality and hierarchy constraint.

**Example 1.** Figure 1 is a class diagram that describes three classes named *Advisor*, *MasterStudent* and *PhDStudent*, one association named *Advise* between instances of the *Advisor* and the *MasterStudent* classes with roles named *advisor* and *ma*, respectively, a cardinality constraint that is imposed on this association, and two class hierarchy constraints between *MasterStudent* to *Advisor* and between *PhDStudent* to *Advisor*. The cardinality constraint states that every *Master* student must be advised by exactly one *Advisor*, while every *Advisor* must advise exactly two *Master* students. The class hierarchy constraints state that *Master* students and *PhD* students are *Advisors* as well, implying that the *Advisor* of a *Master* student can be a *Master* student or a *PhD* student.



**Fig. 1.** A Class Diagram that is Strongly unsatisfiable

However, in the presence of constraints, a model diagram may turn vacuous as it might impose constraints that cannot be satisfied. For class diagrams, that means that no intended world can satisfy all constraints simultaneously. Consider Figure 1, the interaction between the cardinality constraint of the *Advise* association and the class hierarchy causes an inconsistency problem. Every *Master* student must have a single *Advisor*, while each *Advisor* should advise two *Master* students. However, since the *MasterStudent* class specializes the *Advisor* class, every *Master* student must advise two *Master* students which have each

a single *Advisor*. In order to satisfy this constraint, the *Master* class must be either empty or infinite. Since the intention behind class diagram is to describe non-empty and finite classes, this diagram does not admit any intended world. Such diagrams are termed strongly unsatisfiable.

The problem of satisfiability has been studied by several authors in the context of various kinds of conceptual schemata [1],[3], [6],[7],[11], [14]. Methods have been developed for testing strong satisfiability, for detecting causes for unsatisfiability, and for heuristic suggestions for diagram correction. Yet, no method provides a feasible solution for detecting unsatisfiability for the combination of cardinality constraints and class hierarchy constraints that are present in UML2 class diagrams.

In this paper, we present an algorithm for testing the satisfiability of UML class diagrams that include binary associations and class hierarchies with “disjoint/overlapping” and “complete/incomplete” constraints. The algorithm is based on a reduction to the algorithm of Lenzerini and Nobili [11] that was applied only to ER-diagrams without class hierarchies. Our algorithm analyses all possible combinations of class hierarchy constraints, and provide solutions for all cases. The algorithm is simple and feasible since it adds in the worst case only a linear amount of entities to the original diagram. It improves over previous extensions of the Lenzerini and Nobili method that require the addition of an exponential number of new entities to the original diagram [3]. An implementation of our method within a UML case tool is currently under development. The paper is organized as follows: Section 2 summarizes relevant methods for detecting strong unsatisfiability in class diagrams, introduces the GeneralizationSet notion of UML2, and discusses its semantics. In section 3, we present algorithms for testing satisfiability of the above subset of UML2 class diagrams. Section 4 is the conclusion and discussion of future work

## 2 Background

In this section we present background on the two subjects underlying this research: (1) Methods for identifying consistency of Entity-Relationship (ER) diagrams; (2) The UML 2.0 constraints for specification of class hierarchies. In the first subject we focus mainly on the inequalities based method of Lenzerini and Nobili [11] since our research is based on this method. We also describe shortly the method of Calvanese and Lenzerini [3] since it solves some consistency problems that we solve, but require exponential time.

We denote class symbols as  $C_i$ , association symbols as  $A_i$ , and role symbols as  $rn_i$ . The standard set theoretic semantics of class diagrams associates with a class diagram instances in which classes have extensions that are sets of objects that share structure and operations, and associations have extensions that are relationships among class extensions. Henceforth, we shorten expressions like “instance of an extension of  $C_i$ ” by “instance of  $C_i$ ” and “instance of an extension of  $A_i$ ” by “instance of  $A_i$ ”. The cardinality constraint (also termed multiplicity constraint) imposed on a binary association  $A$  between classes  $C_1$  and  $C_2$  with roles  $rn_1$ ,  $rn_2$  is symbolically denoted:

$$A(rn_1 : C_1[ \min_1, \max_1 ], rn_2 : C_2[ \min_2, \max_2 ]) \quad (1)$$

The multiplicity constraint  $[ \min_1, \max_1 ]$  that is visually written on the  $rn_1$  end of the association line is actually a participation constraint on instances of  $C_2$ . It states that an instance of  $C_2$  can be related via  $A$  to  $n$  instances of  $C_1$ , where  $n$  lies in the interval  $[ \min_1, \max_1 ]$ . A class hierarchy constraint between a super class  $C_1$  and a subclass  $C_2$  is written  $subtype(C_2, C_1)$ . It states a subset relation between extensions of  $C_2$  and  $C_1$ . For example, in the class diagram presented in Figure 1, there are three class symbols *Advisor*, *MasterStudent*, *PhDSudent*, one association symbol *Advise* with the cardinality constrain:

$$Advise(advvisor : Advisor[1, 1], ma : MasterStudent[2, 2])$$

And two class hierarchy constraints:

$$subtype(MasterStudent, Advisor), subtype(PhDStudent, Advisor).$$

A *legal instance* of a class diagram is an instance where the class and association extensions satisfy all constraints in the diagram. A class diagram is satisfiable if it has a legal instance. However, following [11], it appears that a more meaningful satisfiability property involves intended (desirable) instances, in which all class extensions are finite, and not all are empty. Instances in which all class extensions are empty are termed empty instances. A class diagram is strongly satisfiable if for every class symbol in the diagram there is an instance world with a non-empty and finite class extension<sup>1</sup>. Lenzerini and Nobili also show, for the restricted version of class diagrams that they deal with, that a strongly satisfiable class diagram has an instance in which all class extensions are non-empty and finite. The problem of testing whether a class diagram is strongly satisfiable is called the consistency problem of class diagrams. The standard semantics of class diagrams does not refer to relationships between extensions of class symbols that are not constrained by class hierarchy constraints. That is, class extensions can intersect. Yet, we know that in object-oriented software systems, objects are created using constructors which immediately declare them as objects of certain classes. Therefore, in object-oriented software systems multiple class membership is possible only via the polymorphic semantics of class hierarchies.

## 2.1 Methods for Testing Consistency of Class Diagrams

The method of Lenzerini and Nobili is defined for ER diagrams that include Entity Types (Classes), Binary Relationships, and Cardinality Constraints. The method consists of a transformation of the cardinality constraints into a set of linear inequalities whose size is polynomial in the size of the diagram. Strong satisfiability of the ER diagram reduces to solution existence of the associated

---

<sup>1</sup> The notions of satisfiability and strong satisfiability were introduced by Lenzerini and Nobili [11] with respect to Entity-Relationship diagram.

linear inequalities system. The linear inequalities system denoted  $\psi$ , that is associated with a given ER schema is defined as follow:

1. For each association  $R(rn_1 : C_1[\min_1, \max_1], rn_2 : C_2[\min_2, \max_2])$  insert four inequalities:

$$r \geq \min_2 \cdot c_1, \quad r \leq \max_2 \cdot c_1, \quad r \geq \min_1 \cdot c_2, \quad r \leq \max_1 \cdot c_2$$

2. For every entity or association symbol  $T$  insert the inequality:  $T > 0$

Lenzerini and Nobili also present a method for identification of causes for non-satisfiability. This method is based on a transformation of the conceptual schema into a graph and identification of critical cycles. Similar approaches are introduced also in the works of [14], [6]. Hartman in [7] further develops methods for handling satisfiability problems in the context of database key and functional dependency constraints. Heuristic methods for constraint corrections are presented in [7], [8].

Calvanese and Lenzerini in [3] elaborate the inequalities based method of [11] to apply to schemata with class hierarchy (also called ISA) constraints. They provide a solution to the problem of testing strong satisfiability for class diagrams in the presence of hierarchy constraints. The expansion is based on the assumption that class extensions may overlap. They provide a two stage algorithm in which the consistency problem of a class diagram with ISA constraints is reduced into the consistency problem of a class diagram having no class hierarchy constraints. Then similarly to what is done in [11], the authors develop a method for checking the satisfiability of the new class diagram by deriving a special system of linear inequalities. However, compared to [11], they suggest a different technique for deriving the inequalities system.

The class diagram transformation process of [3] is fairly complex, and might introduce, in the worst case, an exponential number of new classes and associations, in terms of the input diagram size. The method of [3] was further simplified by [4], where class overlapping is restricted to class hierarchy alone. The simplification of [4] reduces the overall numbers of classes and association compared to [3], but it still introduces, in the worst case, an exponential number of new classes and associations. Example 2 below presents the application of [4] to Figure 1.

**Example 2.** After applying [4] method we have four different classes and eight distinct specializations for the *Advise* association. Each class and association is represented by a variable as follow:

$a_1$  for an *Advisor* that is neither a *Master* nor a *PhD*;  $m_2$  for an *Advisor* that is a *Master* but not a *PhD*;  $p_3$  for an *Advisor* that is a *PhD* but not a *Master*;  $mp_4$  for an *Advisor* that is simultaneously a *Master* and a *PhD*;  $\{ad_{ij} | 1 \leq i \leq 4 \wedge j \in \{2, 4\}\}$  for the association variables

Every specialized association relates two new classes, one for the advisor role and the another for the *ma* role. The indexes represent the index of the class variable. For example the variable  $r_{12}$  represents the specialization of the *Advise* association to an association between *Advisors* who are neither *Masters* nor *PhD*

student (the  $a_1$  variable) and *Advisors* specialized to *Masters* but not to *Ph.Ds* (the  $m_2$  variable). Below the [4]-inequalities system for the class diagram in Figure 1. Equations 1-4 translate the 2..2 multiplicity, equations 5-6 translate the 1..1 multiplicity, and the equations in 8 represent the satisfiability conditions. The resulting inequalities system that includes the inequalities 1-8 is an unsolvable; therefore the class diagram in Figure 1 is inconsistent.

1.  $2a_1 = ad_{12} + ad_{14}$
2.  $2m_2 = ad_{22} + ad_{24}$
3.  $2p_3 = ad_{32} + ad_{34}$
4.  $2mp_4 = ad_{42} + ad_{44}$
5.  $m_2 = ad_{12} + ad_{22} + ad_{32} + ad_{42}$
6.  $p_4 = ad_{14} + ad_{24} + ad_{34} + ad_{44}$
7.  $a_1, m_2, p_3, mp_4, ad_{12}, ad_{14}, ad_{22}, ad_{24}, ad_{32}, ad_{34}, ad_{42}, ad_{44} \geq 0$
8.  $a_1 + m_2 + d_3 + md_4 > 0$ , and  $ad_{14} + ad_{22} + ad_{24} + ad_{32} + ad_{34} + ad_{42} + ad_{44} > 0$

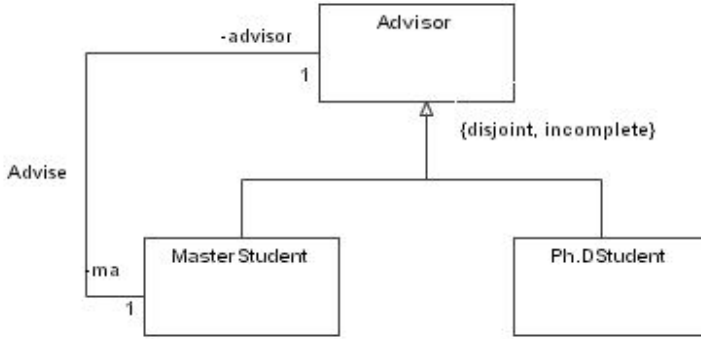
The main contribution of [3] is the presentation of an effective method for reasoning about ISA and cardinality constraints. They present a sound and complete method for logical implication of cardinality or ISA constraints. Yet their method requires, in the worst case, the addition of an exponential number of variables and inequalities). The authors of [3],[4] does not take into consideration type features of the schemata, such as disjunctive subtyping.

## 2.2 UML2.0 Class Hierarchy Concepts: Generalization Sets

In UML2.0 class hierarchy constraints are expressed using the Generalization-Set concept. It seems that a GeneralizationSet construct is similar to the older class hierarchy grouping construct that could be associated with a discriminator. The UML2.0 text for GeneralizationSet specification is: "The GeneralizationSet association designates the partition to which the Generalization link belongs. All of the Generalization links that share a given general Classifier are divided into disjoint sets (that is, partitions) using the GeneralizationSet association. Each partition represents an orthogonal dimension of specialization of the general Classifier" [12]. A GeneralizationSet may be labeled with one of following constraint tags:

1. *complete, disjoint* - Indicates that the partition covers the super class, and the subclasses are disjoint.
2. *incomplete, disjoint* - Indicates that the partition does not cover the super class, and the sub classes are disjoint.
3. *complete, overlapping* - Indicates that the partition covers the super class, and the subclasses are overlapping.
4. *incomplete, overlapping* - Indicates that the partition does not cover the super class, and the subclasses are overlapping.

The GeneralizationSet constraints in Figure 2 mean that: (1) there is at least one *Advisor* who is neither a *Master* nor a *PhD* (2) there is no *Advisor* who is both a *Master* and a *PhD*. In the next section we provide solutions for testing strong satisfiability in the presence of generalization set constraints.



**Fig. 2.** Generalization Set Constraint

### 3 Testing Consistency of UML2.0 Class Diagrams That Include Generalization Sets

In this section, we present a method for testing the satisfiability (and thereby for strong satisfiability) of class diagrams that include UML2.0 generalization set constructs. We provide solutions to the four UML2.0 generalization set constraints, as described in Section 2.2. The method builds on top of the Lenzerini and Nobili [11] algorithm, which translates a restricted ER diagram into an inequality system and tests for the existence of a solution. For the simplest case of an unconstrained generalization set construct, we reduce the consistency problem of a class diagram into the consistency problem of a class diagram that is equivalent to the [11] ER diagram. For the cases of constrained generalization sets, the consistency problem of a class diagram is reduced into the consistency problem of a constrained class diagram without class hierarchy, for which the method of [11] can be applied with small extensions. First, we state that satisfiability implies strong satisfiability also in the presence of generalization sets. The proof is similar to the proof for the case of the restricted ER diagrams, as presented in [11].

**Theorem 1.** *If a class diagram with possibly constrained generalization sets is strongly satisfiable, then it has an instance in which all class extensions are non-empty and finite.*

#### 3.1 Testing the Consistency of Class Diagrams with Unconstrained Generalization Sets

##### Algorithm 1

- **Input:** A class diagram CD that includes binary associations and unconstrained generalization sets.
- **Output:** True, if the CD is satisfiable; false if CD is not satisfiable.

– **Method:**

1. Class diagram reduction - Create a new class diagram  $CD'$  as follow
  - (a) Initialize  $CD'$  by the input class diagram  $CD$ .
  - (b) Remove from  $CD'$  all generalization set constructs.
  - (c) For every removed generalization set construct create new binary associations between the superclass to the subclasses, with 1..1 participation constraint for the subclass (written on the super class edge in the diagram) and 0..1 participation constraint for the super class.
2. Apply the Lenzerini and Nobili algorithm to  $CD'$ .

**Example 3.** Figure 3 is the reduced class diagram of Figure 1, following step 1 in the algorithm. Applying the inequalities method of [11] (step 2) yields the inequalities system presented below. We describe the inequalities system for Figure 3 using the symbols  $a$  for *Advisor*,  $m$  for *Master*,  $p$  for PhD,  $ad$  for *advise* and  $x, y, \cdot$  for the new associations *ISA1*, and *ISA2* respectively.

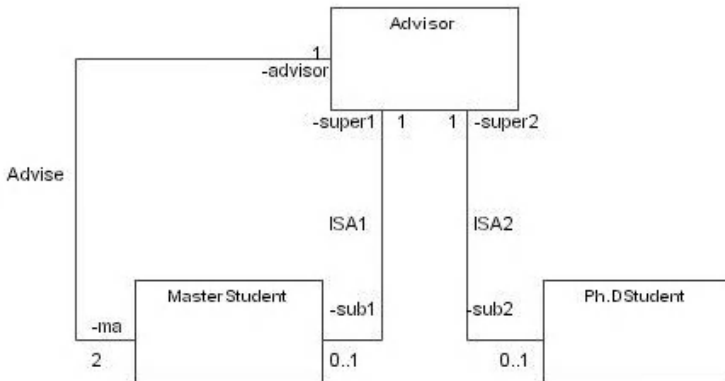
$$ad \geq 2a, ad \leq 2a, ad \leq m, ad \geq m, x \geq m, x \leq m, x \leq a, y \geq p, y \leq p, y \leq a$$

This system has no solution and therefore the [11] algorithm returns False. The same result was obtained in Section 2 by applying the [3],[4] algorithm to Example 1.

**Claim 1:** Algorithm 1 tests for satisfiability of class diagrams with unconstrained generalization sets.

*Proof.* (sketched) the claim builds on showing that the translated class diagram  $CD'$  preserves the satisfiability of the input class diagram  $CD$ . Full proof appears in [10]

**Claim 2:** Algorithm 1 adds to the [11] method an  $O(n)$  time complexity, where  $n$  is the size of the class diagram (including associations, classes and ISA constraints).



**Fig. 3.** The Reduced Class Diagram of Figure 1

*Proof.* The additional work involves the class diagram reduction, which creates a class diagram with the same set of classes and one additional association that replaces every class hierarchy constraint. Since there is a linear additional work per generalization set, the overall additional work is a linear to the size of the class diagram.

*Note 1.* The application of our method to Example 1 requires the addition of two associations to the reduced class diagram. The resulting inequalities system has six variables and fifteen inequalities. In comparison, the application of the [3], [4] methods to Example 1 results a class diagram with 4 classes and 8 additional associations. The resulting inequalities system has twelve variables and twenty-six inequalities.

### 3.2 Testing the Consistency of Class Diagrams with Constrained Generalization Sets

In order to test satisfiability under constraints we extend the inequalities system obtained by the previous Algorithm with inequalities that guarantee the existence of a world instance that satisfies the constraint. The additional inequalities do not provide an equivalent characterization for the required constraint, but are sufficient for reducing the satisfiability problem.

#### Algorithm 2

- **Input:** A class diagram CD that includes binary associations and UML2.0 constrained generalization sets
- **Output:** True, if the CD is satisfiable; false if CD is not satisfiable.
- **Method:**
  1. Class diagram reduction
    - (a) Steps 1.a, 1.b, 1.c from Algorithm 1
    - (b) Associate the constraint *Const* with the set of classes  $C, C_1, \dots, C_n$  where *disjoint/overlapping* means that there is no/(at least one) instance of class  $C$  who is associated with more than one instance from  $C_1, \dots, C_n$  via the new associations, *complete/incomplete* means that all/part of the instances of class  $C$  are associated with some instances in classes  $C, C_1, \dots, C_n$  via the new associations respectively.
  2. Inequalities system construction
    - (a) Create the inequalities system for CD' according to the Lenzerini and Nobili algorithm.
    - (b) For every constraint *Const* added in step 1b ,extend the inequalities system, as follows:
      - i.  $Const = \text{disjoint, incomplete}: C > \sum_{j=1}^n \sum C_j$
      - ii.  $Const = \text{disjoint, complete}: C = \sum_{j=1}^n \sum C_j$



iii. *Const = overlapping, complete*:  $C < \sum_{j=1}^n \sum C_j$

iv. *Const = overlapping, incomplete*:  $\forall j \in [1, n]. C > C_j$

**Example 4.** Consider Figure 3. The interaction between the cardinality constraint, hierarchy, and the generalization set constraints causes an inconsistency problem. Applying the method of [11] with the extension in Algorithm 2, step 2.b.i to the reduced class diagram of Figure 3 yields the insolvable inequalities system presented below. Therefore the class diagram in Figure 3 is inconsistent.

$$x = m, x \leq a, y = d, y \leq a, ad = d, ad = m, ad > 0, d > 0, m > 0, p > 0, \\ x > 0, y > 0, \text{ and the added inequality } ad > m + p$$

**Claim 3:** Algorithm 2 tests for satisfiability of class diagrams with UML2.0 constrained generalization sets.

*Proof.* (sketched) The claim builds on showing that the translated class diagram CD' preserves the satisfiability of the input class diagram CD. Full proof appears in [10]. For each constraint we show that the additional inequality (or equality) provides a necessary and sufficient condition for the existing of an instance that satisfies the generalization set constraint. For example inequality [1] in step 2.b of Algorithm 2 characterizes the existing of instance that satisfies the disjoint, incomplete constraint; of course this does not mean that all instances must be disjoint and incomplete.

**Claim 4:** Algorithm 2 adds to the [11] method an  $O(n)$  time complexity, where  $n$  is the size of the class diagram (including the associations, and classes, ISA constraints).

*Proof.* The additional work involves the class diagram reduction, which creates a class diagram with the same set of classes, one additional association that replaces every class hierarchy constraint and one additional inequality for every generalization set constraint. Since there is a linear additional work per generalization set, the overall additional work is a linear to the size of the class diagram

### 3.3 Discussion of the Semantics of the Overlapping and Disjoint Constraints

The overlapping/disjoint semantics in UML documents is confusing and may have different interpretations. The ambiguous issue involves the specification of exactly which subclasses are required to be overlapping/disjoint. The reasonable interpretations for each case involve either the overall collection of subclasses, or a pairwise restriction on subclasses. Correspondingly, we suggest two feasible interpretations for each kind of restriction:

1. Local Overlapping: Meaning there exists an overlapping pair of subclasses in a generalization set.
2. Global Overlapping: Meaning non-empty intersection of all subclasses in a generalization set.

3. Local Disjointness: Meaning every pair of subclasses in a generalization set is disjoint.
4. Global Disjointness: Meaning empty intersection of all subclasses in a generalization set.

For overlapping constraints, the global version is more restrictive, i.e., implies the local version, while for disjointness constraints, the local version is more restrictive, i.e., implies the global version. We think that both versions are reasonable and might be necessary, although the local version might be more popular for both constraints [13]. Therefore, we suggest to adopt the local version for each kind of constraint as its default, and require explicit specification if the global version is intended.

## 4 Conclusions and Future Work

In this paper, we introduce a simple and effective algorithm for checking the satisfiability of class diagrams that Include ISA hierarchy with generalization set constraints, following of the semantic of UML 2.0. The simplicity and effectiveness of the algorithm make the implementation straightforward. It is provide a novel improvement over existing methods that require exponential time in the worst case.

In the future, we plan to extend our algorithm to handle the global disjointness, and global overlapping. We also intend to extend our techniques for detecting reasons for inconsistency and develop efficient techniques for possible corrections. The work of [7], [9] presented a number of heuristic strategies for repairing an inconsistent constraint in ER models. We would like to apply similar strategy for reperiing inconsistency in UML 2 class diagram with class hierarchy constraints. Additional extensions will involve the extension of our methods for testing inconsistency in the presence n-ary association with complex cardinality constraints.

## References

- [1] Boufares, F., Bennaceur, H.: Consistency Problems in ER-schemas for Database Systems. Information Sciences, Issue 4 (2004)
- [2] Berardi, D., Calvanese, D., Giacomo, De.: Reasoning on UML class diagrams. Artificial Intelligence (2005)
- [3] Calvanese, D, Lenzerini, M. On the Interaction between ISA and Cardinality Constraints. Proc. of the 10th IEEE Int. Conf. on Data Engineering (1994)
- [4] Cadoli, M, Calvanese, D, De Giacomo, G, Mancini, T, Finite Satisfiability of UML Class Diagrams by Constraint Programming. In Proc. of the CP 2004 Workshop on CSP Techniques with Immediate Application, (2004)
- [5] Liang, P, Formalization of Static and Dynamic UML Using Algebraic. Master's thesis, University of Brussel (2001)
- [6] Hartman, S, Graph Theoretic Methods to Construct Entity-Relationship Databases. LNCS, Vol. 1017, (1995)

- [7] Hartman, S, On the Implication Problem for Cardinality Constraints and Functional dependencies. *Ann.Math.Artificial Intelligence*, (2001)
- [8] Hartman, S, Coping with inconsistent constraint speciation., *LNCS*, Vol 2224, (2001)
- [9] Hartman, S, Soft Constraints and Heuristic Constraint Correction in Entity-Relationship Modeling, *LNCC*, Vol. 2582, (2002)
- [10] Maraee, A.: Consistency Problems in UML Class Diagram. Master' thesis, Ben-Gurion University of the Negev (2006)
- [11] Lenzerini, M, Nobili, P, on the Satisfiability of Dependency Constraints in Entity-Relationship Schemata, *Information Systems*, Vol. 15, 4, (1990)
- [12] OMG, UML 2.0 Superstructure Specification, (2004)
- [13] Rumbaugh, J, Jacobson, G, Booch, G, The Unified Modeling Language Reference Manual Second Edition, Adison Wesley (2004)
- [14] Thalheim, B, Entity Relationship Modeling, *Foundation of Database Technology*, Springer-Verlag, (2000)

# A Framework-Based Design for Supporting Availability of Layered Distributed Applications<sup>\*</sup>

Heung Seok Chae<sup>1</sup>, Jaegeol Park<sup>1</sup>, Jinwook Park<sup>1</sup>, and Sungho Ha<sup>1</sup>,  
and Joon-Sang Lee<sup>2</sup>

<sup>1</sup> Department of Computer Science and Engineering, Pusan National University,  
Busan 609-735, Republic of Korea

{hschae, jgpark, jwpark, shha}@pusan.ac.kr

<sup>2</sup> LG Electronics, Republic of Korea

joon@se.kaist.ac.kr

**Abstract.** Availability indicates that a system continuously provides a useful service at any given time. With the success of Internet-based business, there is more and more interest in the availability of systems. This paper presents the Availability Management Framework(AMF), which supports the management of availability for applications in layered distributed systems. We focus on the AMF's flexibility in order to support various failure detection and recovery mechanisms that are required by each application. In addition, this paper describes the result of application of the AMF to support availability of RFID systems.

## 1 Introduction

As new technologies enable powerful new services, we should be confident that a system will operate as we expect and the system will not fail in normal use. Along with reliability, safety, and security, availability is one of the four principal dimensions to dependability of systems [1]. Informally, the availability of a system is the probability that it will be up and running and able to deliver useful services at any given time. There are two measures concerning availability: Mean Time Between Failure (MTBF) and Mean Time To Repair (MTTR). MTBF is the average time between observed system failures. Once a system fails, an important related concept is the time it takes to repair the system. MTTR is a measurement of the time required to bring a system back on line when a failure has occurred. This is often called downtime.

Computer systems vary widely in their tolerance of downtime. Some systems cannot handle even the briefest interruption in service without catastrophic results, whereas others can even handle extended outages while still delivering on their required returns on investment [2]. In other words, there is a whole range of possible availability levels that range from an absolute requirement of 100 percent down to a low level, where it just does not matter if the system

---

<sup>\*</sup> This work was supported by the Regional Research Centers Program(Research Center for Logistics Information Technology), granted by the Korean Ministry of Education & Human Resources Development.

is running or not. These various levels of availability requirement imply that some appropriate failure detection and recovery mechanisms should be selected according to the level of availability.

To address these problems, we propose a framework based approach to supporting flexible management of availability for distributed systems. A framework is a partial, common design and implementation which can be extended to be reused for several specific applications [3]. As one of the widely used techniques in object-oriented paradigm, frameworks have been considered promising technique for reusing the already developed softwares. This is supported by the evidence that there are many frameworks for various domains [4, 5, 6, 7, 8].

The proposed framework provides a flexible availability management by supporting various failure detection and recovery mechanisms that are specific to the systems. The framework embeds failure detection and recovery codes which are common to various applications and provides extension points by which the framework can be extended to accommodate the specific availability requirements of each application.

The rest of the paper is organized as follows: Section 2 makes a brief description of availability and framework-based approach to the availability. Section 3 presents the proposed framework for supporting availability for distributed systems. Section 4 describes a case study with an RFID middleware. The related works and conclusion are given in Section 5 and 6, respectively.

## 2 Backgrounds

### 2.1 Availability

As mentioned above, availability is defined in terms of MTBF and MTTR. In order to increase the availability, we should increase the MTBF and/or decrease the MTTR. In general, the MTBF is concerned with reliability and it is the MTTR that researches on availability make efforts on. For reducing the MTTR, it is necessary to automatically detect and repair failures. Thus, failure detection and failure recovery are the main activities for providing highly available systems.

For prompt recovery of system failures, the first action is to recognize the occurrence of them. There are typically two approaches to detecting failures in distributed systems: *Ping/echo* and *Heartbeat*. In the case of ping/echo, one component (so called, monitoring component) issues a ping and expects to receive back an echo, within a predefined time, from the component under scrutiny (so called, monitored component). In the case of heartbeat ( or dead man timer), the monitored component emits a heartbeat message periodically and the monitoring component listens for it. If the heartbeat fails, the monitored component is assumed to have failed.

The failed service should be recovered in order to provide a continuous service. Failure recovery includes any actions taken to restore the system to service from failure. These actions can cover a wide range of activities from restart of the failed service to failover to other available service. The availability management framework should contain the knowledge of the appropriate recovery action for

the failure of each managed service in the system, whether it involves restoring that service to full operation or switching over to a redundant service.

## 2.2 Application Frameworks

Object-oriented application frameworks are a promising technology for reifying proven software designs and implementations in order to reduce the cost and improve the quality of software [3]. Although there is subtle difference among the definitions of frameworks, the following definition is commonly used: a framework is a reusable, semi-complete application that can be specialized to produce custom applications [7, 9]. In contrast to earlier object-oriented reuse techniques based on class libraries, frameworks are targeted for particular business units and application domains. Frameworks like MacApp, Interviews, ACE, RMI, and implementation of OMG's CORBA play an increasingly important role in contemporary software development [7]. Since its conception at the end of the 1980s, the appealing concept of object-oriented frameworks has attracted attention from many researchers and software engineers. Frameworks have been defined for a large variety of domains, such as user interfaces [10], operating systems [4], financial systems [6], groupware frameworks [11], and frameworks for mobile agents [12].

In order to avoid some misunderstanding of framework concepts, we follow the notion of framework concepts proposed in [3]: core framework design, framework internal increment, application-specific increment, and application. The *core framework design* describes the typical software architecture for applications in a given domain. The core framework design consists of both abstract and concrete classes in the domain. The concrete classes are intended to be invisible to the framework user. On the contrary, an abstract class is generally intended to be subclassed by the framework user to support the variation which is specific to each application. This case is also referred to as *hot spots* [13]. The *framework internal increment* captures common implementations of the core framework design. The internal increment may have two forms: subclasses representing common realizations of the concepts captured by the superclasses, or a collection of subclasses representing the specifications for a complete instantiation of the framework in a particular context. The *application-specific increment* means new classes needed to be developed to fulfill the actual application requirement. That is, the application-specific increment implements those parts of the application design not covered by the core framework design and the internal increment. An *application* is composed of one or more core framework designs, each framework's internal increments, and an application-specific increment.

Availability management is very common feature for applications, especially for distributed softwares. The two concepts of failure detection and recovery are common to every applications which are required to be highly available. In addition, the specific methods for failure detection and recovery may depend on the requirement on the availability of each application. Thus, framework is a natural solution to providing availability management service for distributed applications. That is, the common strategy and methods of failure detection and recovery are captured by the core framework design and part of them are

implemented by the framework internal increment. And then, application-specific increment defines some application specific functionality such as the specific way to detect failures in the application and recover from those failures.

### 3 The Availability Management Framework

#### 3.1 The Client-Server Model of the Framework

The Availability Management Framework(abbreviated as AMF) provides the common design and implementation such as registration, failure detection, and failure recovery. Figure 1 shows the high-level structure of the AMF and the roles of the constituents.

The AMF consists of two parts: *AMFSrv* package and *AMFCli* package. The *AMFSrv* is for the availability management server and *AMFCli* for various client applications. That is, the *AMFSrv* monitors the occurrence of failures in clients, determine appropriate recovery strategy, and request the chosen recovery action to each client involved in the failure. The *AMFCli* supports the implementation of client applications that are managed by the AMF. The *AMFCli* implements the generic functions for failure detection and failure recovery actions. Each client can extend the predefined generic failure detection and recovery action by supplying its applications-specific increment. First, a client should be registered in order to be monitored and recovered from failure. Once the client is registered to the AMF Server, the *AMFSrv* request the *AMFCli* to start failure detection. The *AMFCli* invokes the actual failure detection action that is provided by the client. When some failures are detected by the *AMFCli*, they are reported to the *AMFSrv*. The *AMFSrv* chooses an appropriate recovery strategy depending on the situation. And then, the *AMFSrv* requests each involved client's *AMFCli* to perform the appropriate recovery action. The *AMFCli* invokes the actual recovery action that is provided by the client.

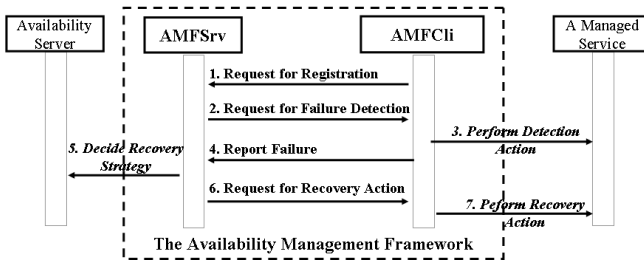


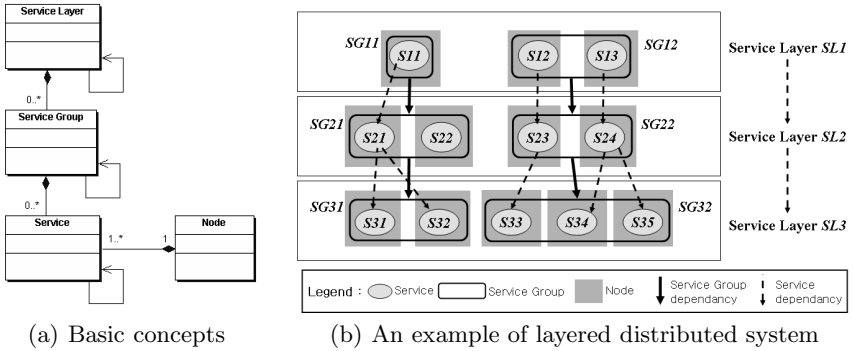
Fig. 1. The Architecture of the AMF

Since the AMF only implements the common features of failure detection and recovery, the actual functions that are specific to each client should be provided by the client and invoked by the framework. The AMF provides three hot spots, the part that should be extended by the framework's clients in order to satisfy the

client's needs. The first is concerned with failure detection. Although the *AMFCli* can provide some typical failure detection mechanisms, the client should implement the actual code for determining the occurrence of failures which naturally depends on the client itself. For example, let us assume that heartbeat mechanism is used for failure detection. Then, the client should provide a code that send a periodic heartbeat to the *AMFCli* and then the *AMFCli* conveys the heartbeat to the *AMFSrv*. The second hot spot is for failure recovery action. As is the case with failure detection, the actual code for failure recovery action depends on the client application. Thus, the failure recovery action should be provided by the client. For example, when the *AMFSrv* requests the client to restart itself via *AMFCli*, the client only knows how to restart itself. The third hot spot is for the selection of recovery strategy by the server. The *AMFSrv* implements some typical recovery strategies such as fail-over and repair. However, it may be necessary for the server to have its own recovery strategy. In that case, the recovery strategy part of *AMFSrv* could be extended by the server.

### 3.2 The Conceptual Model of Availability Management

This section presents the conceptual model and the state model of logical entities that are managed by the AMF. The class diagram in Figure 2 (a) shows the logical entities managed by the AMF and the relationships between them.



**Fig. 2.** The conceptual model of the AMF

A *service* is the smallest unit on which the AMF performs failure detection and recovery. A service can be associated with several different states depending on the characteristics with respect to its availability. Typically, a service has two different states: Active and Failed. A service of active state provides the intended function and a service of failed state cannot perform its function due to some kind of failures. We will present in more detail the state model for a service in Figure 4.

In distributed systems, a service is associated with other services to provide meaningful operation requested by users. In this paper, service *S2* is defined to depend on service *S1* when service *S1* is required for successful operation of service *S2*. For example, let us consider a web browser in a PC and DNS service in DNS



server for two services. The DNS service is required for the web browser to visit a site named by URL. Thus, service *web browser* depends on service *DNS*.

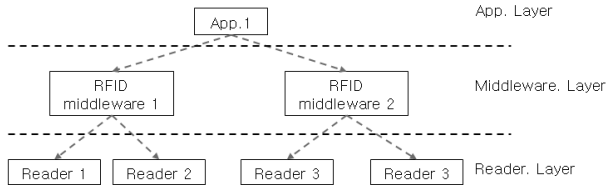
A *service group* is a cluster of the equivalent services; that is, all the services in a service group provide the same behavior to the request from clients. Each service belongs to only one group. A service group can have one of the *Active*, the *Partially active*, or the *Failed* states, depending on the states of the services that belong to the group. The concept of service group is intended for failure recovery strategy in such a way that an available service in the same group is chosen as a take over in stead of a failed service. There is a dependency between two service groups. A dependency between service groups indicates that there can be dependency between the services that belong to the services groups. For example, assuming that service group *SG1* is defined to depend on *SG2*, a service in *SG1* can depend on a service in *SG2*.

For example, consider a configuration of distributed systems in Figure 2 (b). There are six service groups, named *SG11*, *SG12*, ..., *SG31*, *SG32*, each of which consists of one, two, or three services. There are four dependencies between service groups, which is denoted by solid line with arrow between the corresponding rectangle with rounded corner. It is valid for service *S21* to depend on services *S31* and *S32* because service group *SG21* depends on *SG31*. However, it is not allowed for *S21* to depend on services outside group *SG31* such as *S33* or *S34*.

A *service layer* is a cluster of service groups. Each service group belongs to only one layer. There is a dependency between two service layers, which enforce that dependencies between service groups could be defined in a layered fashion. Consider the configuration in Figure 2 (b) where three service layers *SL1*, *SL2* and *SL3* are defined, and *SL1* depends on *SL2* and *SL2* on *SL3*. Because *SL1* depends on *SL2*, a service group (i.e., *SG11* and *SG12*) in *SL1* can have dependency on any service group (i.e., *SG21* and *SG22*) in *SL2*. However, a service group cannot have dependency on a service group in the same layer or in upper layers.

Similar to states of a service group, the state of a service layer is defined to be *Active*, *Partially active*, or *Failed* depending on the states of its composing service groups. A *node* is a processing unit which can run services deployed on it. As seen in Figure 2(a), a node can have multiple services. For simplicity, there is one service on a node in Figure 2(b).

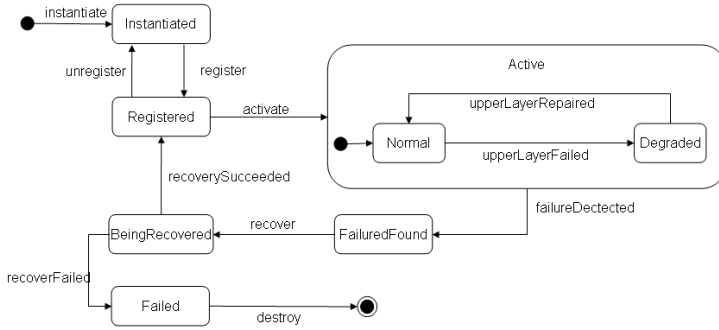
The concept of service layer and group is useful for availability management of large-scale distributed system of multiple homogenous services. In particular,



**Fig. 3.** A simple layered distributed system

the concept of service layer is introduced to simplify the definition of the dependency between a lot of common services in distributed systems. Especially, service layers are very useful when distributed systems are configured in a regular way such as layered configuration or hierarchical configuration. For example, consider RFID systems (see Figure 3). In general, an RFID system consists of three components: readers, RFID middlewares, and application servers. And there exist dependencies between each of them. An RFID middleware depends on operations of many readers, and an application servers can operate properly only if they collect tag data from the RFID middlewares. Although there are a number of readers, RFID middlewares, and application servers, the configuration can be simply captured by the concept of service layer from the viewpoint of availability. That is, a reader layer is positioned at the bottom, an RFID middleware layer is on the reader layer, and finally an application layer is on the RFID middleware layer.

In addition, the concept of dependency between service layers supports the degraded operation of a service when the dependent services at the upper layer are failed. The concept of the degraded operation is captured in the state model of a service. Figure 4 shows the state model of a service in UML state diagram.



**Fig. 4.** The state model of a service

An instantiated service is registered into the AMF as an entity for availability management. By activating the registered service, the service starts to provide service (*Active* state). When a failure is detected by the AMF during the operation of a service, the state of the service is changed into *FailedFound* and the AMF attempts to recover the failed service (into *BeingRecovered* state). The state of the service is changed into *Registered* or *Failed* depending on the success of the recovery. The *Active* state is refined into *Normal* and *Degraded* states, which capture the degraded operation of a service due to failures of dependent services. Initially, the active service is operated under the state of *Normal*, which indicates that all its dependent services at the upper layer are successfully operating. When the AMF detects a failure in some dependent services, it changes the service into *Degraded* state. During the *Degraded* state, the service should run

a special model. For example, consider *Reader1* of an RFID system in Figure 3. When a failure is detected with the *Reader1*, the AMF just attempts to recover by restarting the reader. However, when its dependent service at the upper layer, *RFID middleware1*, fails, the reader cannot send the information collected from RFID tags. Therefore, the AMF changes the state of the reader into *Degraded* so that it can perform an appropriate action, i.e., storing the collected information until the *RFID middleware1* is recovered.

### 3.3 The Support of Failure Detection and Recovery Actions

It is important to support various detection and recovery mechanisms since the selection of detection and recovery actions depends on the requirement on availability for each application. Therefore, the AMF is designed to provide a general framework for failure detection and recovery mechanisms.

Figure 5 shows a class diagram that illustrates the part of the AMF with respect to failure detection mechanisms. The AMF supports four kinds of failure detection mechanisms: passive detection, external active detection, internal active detection, and user-defined detection.

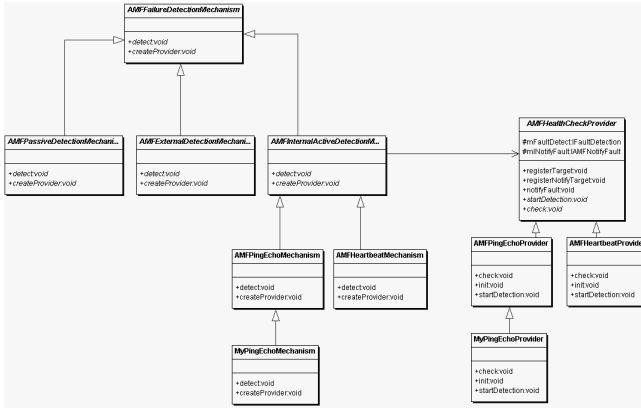


Fig. 5. Design for failure detection mechanism

In the case of passive detection (*AMFPassiveDetectionMechanism*), the service is not involved in the detection, and mostly operating system features are used to decide the health of a service. This includes monitoring the death of processes comprising the service. In external active detection (*AMFExternalDetectionMechanism*), there is no need to include special code for detecting the health of a service. Instead, the AMF examines its health by invoking the service and checking that the service operates in a timely manner. In internal active detection (*AMFInternalActiveDetectionMechanism*), the service can provide the code for checking its own health for more precise and correct detection. The AMF utilizes this code to investigate the health of the service. The AMF

basically provides two detection mechanisms of this kind: *AMFPingEchoMechanism* and *AMFHeartbeatMechanism*. *AMFHealthCheckProvider* is used as a placeholder to indicate the audit code for monitoring the health of the service. Two providers *AMFPingEchoProvider* and *AMFHeartbeatProvider* are already provided as framework internal increment. Developer can define his own detection mechanism that is specific to his application. User-defined detection mechanism can be designed by extending the provided mechanisms; by subclassing any of classes *AMFPassiveDetectionMechanism*, *AMFExternalDetectionMechanism*, *AMFInternalActiveDetectionMechanism*.

As the actual code for monitoring its state is provided by the service, the recovery action naturally depends on each application. To support application-specific recovery action, the AMF enables an application to implement its recovery action depending the state of the service. This is very general and flexible approach since the exact recovery action varies with applications. For example, when a failure is detected in an RFID reader, the AMF will attempt to recover by restarting the reader. At this time, the AMF can only make a restart request to the reader and only the reader knows how to restart itself and can provide the code for restarting.

In the light of this approach, the AMF defines recovery action model based on the state model of Figure 4. *AMFRecoveryProvider* sends a recovery request message to a generic *AMFServiceState* class. The *AMFServiceState* is specialized into several classes that are specific to each state of a service. Depending on its specific recovery strategy, developers can implement the required recovery action by subclassing the appropriate class.

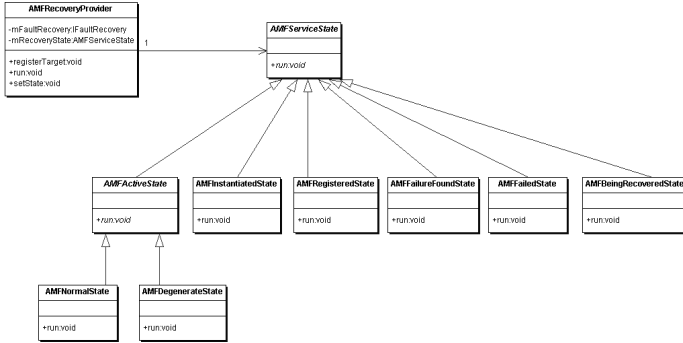


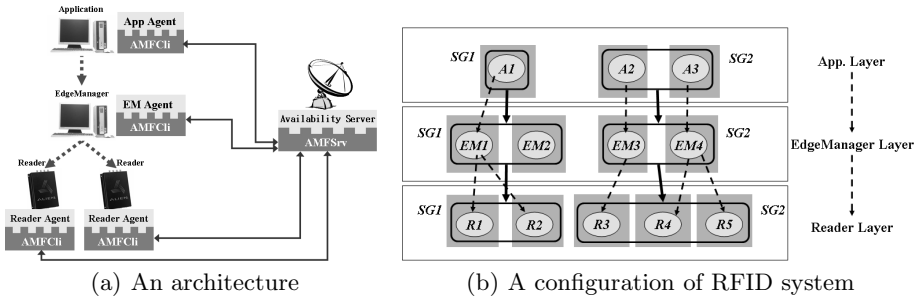
Fig. 6. Design for failure recovery mechanism

## 4 Case Study

This section describes the result of a case study where the proposed AMF is applied to implement availability of RFID system [14]. RFID is a generic term for technologies that use radio waves to automatically identify individual items.

By identifying and tracking each product using RFID tags and readers, it is possible to collect accurate, immediate information about the location of items and the history of items. Due to the automatic identification and collection of information, the RFID system can be used for various applications such as retail, healthcare, logistics, automotive, food industry, etc.

Our research center, Research Center for Logistics Information Technology (for short, LIT), has been developing the core components of RFID systems such as RFID tags, RFID readers, and RFID middlewares. In order to provide availability of RFID systems, we implemented failure detection and recovery functions for RFID readers and RFID middlewares using the proposed AMF. Figure 7 shows an architecture of the RFID systems from the viewpoint of availability.



**Fig. 7.** An application of the AMF to RFID systems

The developed RFID system consists of three components: applications, EdgeManagers, the RFID middleware developed by LIT, and readers. All of them are the target of the availability server that monitors the occurrence of failures and attempt to perform appropriate recovery actions. The availability server has been implemented on the top of the *AMFSrv* package of the AMF that provides a basic functions such as node/service management and recovery strategy selection. Application, EdgeManager, and Reader are the entities that are managed by the AMF and thus each of them is implemented using the *AMFCLI* package of the AMF where some basic functions of failure detection and recovery classes are implemented. Each client implements some client specific actions such as the exact code for failure recovery action and for failure detection action (in the case of internal active mechanism). These codes are illustrated on the top of the *AMFCLI* such as *App Agent*, *EM(EdgeManager) Agent*, and *Reader Agent*. Figure 7 (b) shows a configuration of RFID system which consists of five readers ( $R1, \dots, R5$ ), four EdgeManagers ( $EM1, \dots, EM4$ ), and three applications ( $A1, \dots, A3$ ). For each type of components, three service layers are defined: *App. Layer*, *EdgeManager Layer*, and *Reader Layer*. Within each layer, two service groups and dependencies between them are defined, and each service group consists of one (*SG1* in *App. Layer*) to three services (*SG2* in *Reader Layer*).

Initially, this configuration is registered and maintained by the availability server. Once registered, all the services are managed by the server. There are

two cases of failure recovery when the *EM1* fails. First, the server attempt to recover by restarting *EM1*. At the same time the server notices that *EM2* in the same group is available at that time, and then the server request *R1* and *R2* to run in a degraded operation since the two readers cannot deliver the collected tag information any more to the failed *EM1*. When requested for a degraded operation, the reader service stores the pending tag information within itself or send it to the availability server. After the successful recovery of *EM1*, the server notifies *R1* and *R2* of the recovery of *EM1* so that they continue to deliver tag information to *EM1* followed by the delivery of the stored tags. Second, if the server fails to recover *EM1*, then it attempt to recover by using *EM2* in the same group. That is, *R1* and *R2* are requested to start a transmission of tag information to *EM2* followed by the delivery of the pending tags.

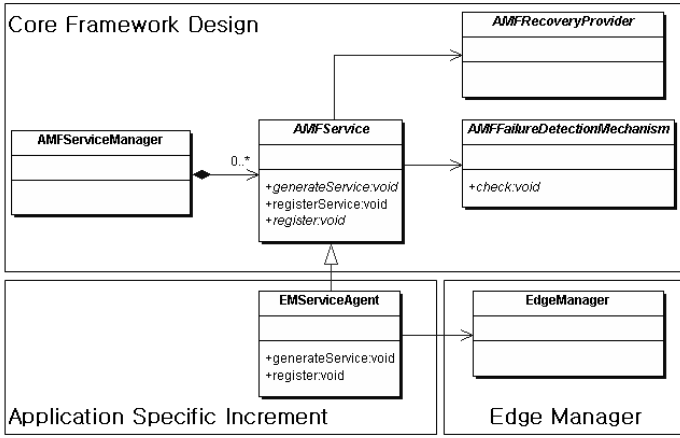


Fig. 8. Registration of EdgeManager as a AMF's Service

Our framework enables developers to use the AMF with ease. For example, Figure 8 shows the class diagram for the design of *EdgeManager Agent*. The upper part is a framework core design, which is designed and implemented by the AMF. Class *EdgeManager*, at the right and bottom position, is EdgeManager itself. Class *EMServiceAgent* is needed to implement as an application specific increment, which provides the actual code for failure detection and recovery actions. For the limitation of space, some classes for the detection and recovery actions such as *MyPingEchoMechanism*, *MyPingEchoProvider*, and *MyRecoveryProvider* are not shown in the class diagram.

## 5 Related Works

The Service Availability Forum is developing Application Interface Standard (AIS)[15] that define programming interfaces to allow applications to be integrated

and coordinated by availability and management middleware. Our framework has the common goal with AIS. However, there is some difference between the AIS and our AMF. First, our framework supports an efficient implementation of availability for applications by using the object-oriented framework technology. Our AMF already designed and implemented a fundamental skeleton codes that are commonly required for available applications. Application programmers just provides the actual code for detection and recovery actions and all other operations are handled by the AMF. In addition, developer can change their detection and recovery action just by subclassing the pre-designed classes in the AMF.

Secondly, our framework supports dependency management for layered distributed systems. Many availability management frameworks and systems such as AIS, CGL [16], SelfReliant [17] just support dependencies between services; their approaches only supports the dependency at the level of services. However, very large-scaled systems such as RFID systems can have some regular configuration such as layered or hierarchical way. Our framework supports a dependency management between service layers where the dependencies between services are constrained by the form of layer, which simplifies the dependency model of services and in addition allows the degraded operation of a service at the failure of an upper layer service.

## 6 Conclusion and Future Work

We have described the high-level design of a framework for supporting availability management of distributed applications. The proposed framework provides flexibility that enables developers to choose appropriate failure detection and recovery methods, depending on availability requirement. In addition, the concept of service layers supports the simplified management of dependencies between services and degraded operation of services. We also performed a case study with an RFID system where the availability of RFID middlewares and readers were efficiently implemented by reusing and extending the framework.

Two research directions are considered future works. First, we plan to enhance our framework by providing framework internal increment for some specific domain such as RFID systems. Second, scalability will be incorporated into the framework. For example, when the AMF determine a take over service instead of a failed service in the same group, it can consider the scalability issue additionally. That is, the AMF can be enhanced to choose the service with the least load among several available services in the same group.

## References

1. Laprie, J.-C.: Dependable Computing: Concepts, Limits, Challenges, In: Proc. of 25th IEEE Symposium on Fault-Tolerant Computing (1995)
2. Marcus, E., Stern, H.: Blueprints for High Availability, Wiley, (2003)
3. Fayad, M., Schmidt, D.: Object-Oriented Application Framework. CACM 40(10) (1997)

4. Campbell, R.H., Islam, N.: A Technique for Documenting the Framework of an Object-oriented System. *Computing Systems* 6(4) (1993)
5. Schmidt, D.C.: Applying Design Patterns and Frameworks to Develop Object-oriented Communication Software. In *Handbook of Programming Languages*, Vol. I (1997)
6. Birrer, E.T.: Frameworks in the Financial Engineering Domain: An Experience Report. In: *Proc. of ECOOP '93* (1993)
7. Fayad, M.E., Schmidt, D.C., Johnson, R.E.: *Object-Oriented Application Frameworks: Problems and Perspectives*. Wiley (1997)
8. Fayad, M.E., Schmidt, D.C., Johnson, R.E.: *Object-Oriented Application Frameworks: Implementation and Experience*. Wiley (1997)
9. Johnson, R.E., Foote, B.: Designing Reusable Classes. *Journal of Object-Oriented Programming* 1(2) (1988)
10. Linton, M.A., Vlissides, J.M., Calder, P.R.: Composing User Interfaces with InterViews. *IEEE Computer* 22(2) (1989)
11. Koch, M., Koch, J.: Application of Frameworks in Groupware - The Iris Group Editor Environment. *ACM Computing Surveys* (2000)
12. Kendall, E.A., Krishna, P., Pathak, C.V.: An Application Framework for Intelligent and Mobile Agents. *ACM Computing Surveys* (2000)
13. Pree, W.: Meta Patterns - A Means for Capturing the Essentials of Reusable Object-Oriented Design. In: *Proc. of 8th ECOOP* (1994)
14. EPCglobal Network, [http://www.epcglobalinc.com/news/EPCglobal\\_Network\\_Overview\\_10072004.pdf](http://www.epcglobalinc.com/news/EPCglobal_Network_Overview_10072004.pdf)
15. SA Forum, Application Interface Specification, SAI-AIS-AMF-B.01.01
16. Carrier Grade Linux, [http://www.osdl.org/lab\\_activities/carrier\\_grade\\_linux](http://www.osdl.org/lab_activities/carrier_grade_linux)
17. SelfReliant, GoAhead Software Inc. <http://www.goahead.com>



# Web Application Security Gateway with Java Non-blocking IO\*

Zhenxing Luo, Nuermainaiti Heilili, Dawei XU, Chen Zhao, and Zuoquan Lin

LMAM, Department of Information Science, Peking University,  
Beijing, 100871, China  
{lzx0728, nur, xudw, czhao, lz}@is.pku.edu.cn

**Abstract.** We present the design and implementation of the WebDaemon Security Gateway (WDSG) with the techniques of event-driving, non-blocking IO multiplexing, secure cookies, SSL and caches based on PKI framework and role-based access control (RBAC) policy. It not only supports massive concurrency and avoids the pitfalls of traditional block I/O based design, but also is able to secure all the resources of an enterprise and reduce the cost and complexity of administration.

## 1 Introduction

With the increasing number of applications running on heterogeneous platforms and distributed information networks, users are bogged down by various logins and forced to remember multiple user identifiers and passwords, which is a hidden danger for users and enterprises. So enterprises need a centralized Integrative Security Management Framework for Web-Based Enterprise Applications. In latest work, we designed and implemented the WebDaemon system [1], an integrative security management solution for Web-based enterprise applications. The WebDaemon allows users access to any authorized applications based on a single authentication that is granted when users firstly access the enterprise web by SSO (Single Sign-On). It provides restricted access to Web-based content, portal and Web applications by role-based access control (RBAC) policies[2] and stronger authentication mechanisms such as digital certificates, tokens, and still adopts the mechanism of secure cookies[3] to avoid the attacking from network, juggling, delaying and replaying, etc. It is able to secure all the resources of the enterprise and reduce the cost and complexity of administration with consistency of policy management.

The WebDaemon Security Gateway (WDSG) is the core module of the WebDaemon, handling security controls by analyzing the HTTP and HTTPS protocols, it runs as a reverse proxy for protected backend servers. Once the user's request is authenticated by WDSG, user can access all the corresponding resources according to his granted authentication. Furthermore, WDSG is the

---

\* This work was supported partially by NSFC (grant numbers 60373002 and 60496322) and by a NKBRPC (2004CB318000)".

unique entry of Web access to all sorts of applications. However, along with the increasing scale of the enterprise application, the performance of WDSG is becoming more critical. And I/O (Input/Output) operation of data is very pivotal factor related to the performance of the server.

Originally, the design of WDSG is based on traditional blocking I/O. The servers were susceptible to enormous thread overhead. Moreover, it still can't handle large-scale users' access effectively. The time required for context switching increases significantly along with the increasing of the number of activated threads, which results in both performance decrease and lack of scalability. The SUN company introduces I/O multiplexing into Java 1.4, which builds on selectors and non-blocking I/O of Java NIO (New I/O) API[4]. In this way, we can improve scalability and performance of the system. Such features are not possible in traditional blocking I/O based design.

In this paper, we present the design and implementation of WDSG. Firstly, we consider the networking aspects such as accepting sockets, reading and writing the data from and to the network, in the implementation of WDSG we use the new traits of the Java NIO available in JDK1.4. Secondly, we implement the access control server with the techniques of event-driving, secure cookies, SSL (Secure Socket Layer) and caches based on PKI framework (Public Key Infrastructure). WDSG provides the Single Sign-On to multiple Web applications and delegates the administration privilege of Web resources to a special module that reduces administration costs through role-based access control. As pointed out in [5], DAC (Discretionary Access Control) or MAC (Mandatory Access Control) can be expressed in RBAC, and they are also alternative access control policies.

The rest of the paper is organized into the following sections. In section 2, we introduce the architecture of WDSG. We show the details of implementation of related secure strategy and cache technique in section 3, respectively. Then, the performance of WDSG is evaluated in section 4. Finally, we summarize the work and point out the further work to continue on this topic.

## 2 Architecture

WDSG adopts the server-pull structure to obtain the user's attributes. User can interact with WDSG through a standard Web browser. After receiving user's request, WDSG determines whether the user has the privilege to access the resource. Once the user's request is permitted, WDSG will retrieve the resource from relevant backend servers according to user's granted authentication and send the content to the user's Web browser. Gateway is transparent to users. WDSG is mainly responsible for decision-making, authorization, distributing resource. The architecture of WDSG is shown in Fig 1.

The acceptor module detects channels of every client to read from or write to the network by a selector class, and the registers build objects into another special running thread. Generally, only one acceptor is enough, if there are several different protocols to operate, we should use one acceptor for each protocol to handle with. It can not only improve operating speed, but also lower the process complexity, strengthen the readability of the procedure and the maintenance of system.

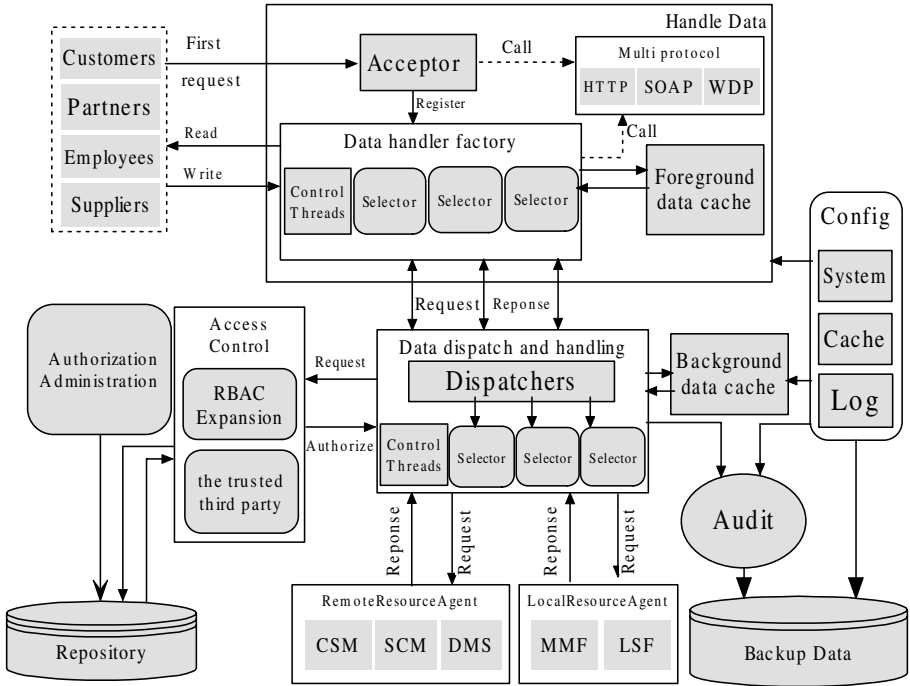


Fig. 1. WDSG architecture

The multi-protocol handling module is used to parse and pack multifarious protocols, for example, HTTP, FTP, HTTPS, UDP, SOAP<sup>1</sup> and WDP (Web-Daemon Protocol [1]). The WDP is a custom protocol to exchange information between access control module (or the trusted third party) and data dispatch module. The custom protocol is simple and fleetly transmitted by UDP, which efficiently use the system resources, taking advantage of an optimized protocol and a Java NIO implementation.

The data handler factory builds on a thread class and a Selector class, it is used to handle NIO objects (not thread) of every client using event-driven. It fulfils reading and writing data from request and response. Both acceptor and data handler factory modules are based upon a thread, which communicates information with other modules by a call back module.

In the foreground and background data cache module, which are mainly built with the buffer class of NIO package, we use caching technique to reduce the delay experienced by the end user. The Web resources frequently requested and retrieved from backend servers and authorization results returned from the access control module are cached, also half-baked data packages and waiting data

<sup>1</sup> World Wide Web Consortium (W3C) (SOAP) 1.2, <http://www.w3.org/TR/SOAP/>, 2004.

packages need be cached. The latter can update cached objects at once when the resources of backend servers are changed.

The Data dispatch and handling module is used to dispatch request context of every client to its respective event handler based on NIO. These object handlers analyze their data independently, which are put into respective thread with an instance of Selector class. In real time, it is important to decide the number of threads beforehand in the module according to the scale of users and the CPU usage, generally, just a few threads are enough to handle all generated objects.

With the techniques of event-driving, non-blocking IO multiplexing and multi-threads, the `local_resource_agent` and `remote_resource_agent` module can autonomously interact with all kinds of the backend servers, obtain corresponding resources and services from relevant local/remote applications respectively, and include a series operation of data packing and unpacking, error handling, etc.

The access control module, based on the event-driven mechanism and multiplexing model, provides the decision making of security policy. It can adopt various security policy, DAC, MAC, RBAC and compatible with XACML (eXtensible Access Control Markup Language<sup>2</sup>). We implemented the policy of RBAC in WebDaemon. It is very important that who the user is, what he/she to do, how he /she to do, all of which are decided through it.

Note that the relations among the access control module, the `local_resource_agent` and the `remote_resource_agent` module, which work simultaneously, that means, whether the user is authorized or not, the two resource agents keep accessing the resources of the backend applications, the junction of both access control and retrieving resource is at the time of preparing to write the data to user's Web browser. There are obvious improvements of performance with this method in practice.

The audit module provides recording every user's trace, caching hit rate and user's authorization, relative information of the procedure, etc. The security administrators can analyze all of the recorded data using our tools, which help them to detect the hackers attacking and some application problems in time. Moreover, with improving security configuration of application, they can close any possible doorway that permits such nefarious attacks to happen.

Authorization Administration module is responsible for assigning roles to actions on the resources, and also assigning users to roles. In the authorization policies, coarse-grain resources like documents are expressed as URL, fine-grain resources like some parts in web pages differ in parameters (POST, GET or HEAD method in HTTP requests) expressed as URL adding these different parameters.

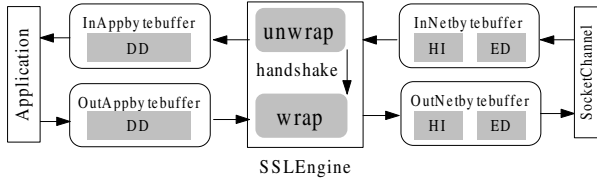
## 3 Related Policies

### 3.1 Security Policy

**SSL.** Firstly, we adopt SSL connection to ensure secure communication between client and WDSG in NIO mode. The connection mode is determined by a secure

---

<sup>2</sup> OASIS, Organization for the Advancement of Structured Information Standards (<http://www.oasis-open.org/>)



**Fig. 2.** The wrap and unwrap are methods of SSLEngine. Encrypted data(ED), handshake information(HI),decrypted data(DD).

flag in order to selectively control what kind of data need to be encrypted. We install a SSL certificate for authentication using public key cryptography. We implement SSL from the latest released JDK1.5 with the introduction of the SSLEngine API [4], which performs all communication with client using byte buffers. We implement the procedure, which includes handshaking, encryption and decryption. Fig 2 simply shows the data flow of the SSLEngine.

**Secure Cookie.** Cookies can contain any arbitrary information the web server chooses, which are used to introduce state otherwise stateless into HTTP transactions. We design and implement the secure cookies to ensure authentication, integrity, and confidentiality[3]. To protect contents of cookies not to be changed by unauthorized modification, a seal cookie is created in a set of cookies, which has the digital signature or MAC (Message Authentication Code) of the server. After a user logs into the web server successfully, WDSG will issue a set of secure cookies to the client of user in real time. If user logs fail, user has to start again by receiving new secure cookies from the web server.

**Certificate.** To satisfy enterprise’s special needs, we use the extension fields of X.509 certificate [6], we separate the authority for attribute-issuing from public-key-issuing, which can be extended multiple times on a basic certificate by different authorities. Note that if the basic certificate is revoked, then all the attributes become invalid. Moreover, the extended certificate is compatible with an X.509. When the user is out of the enterprise network, he/she has to provide the extended X.509 certificate.

**RBAC.** To manage and enforce security in large-scale enterprise application, we adopt the RBAC model in which permissions are associated with roles and users are assigned to appropriate roles. We extend the RBAC96 [7] model with a time characteristic to satisfy the current needs of enterprise, which ensures that only authorized users are given access to certain data, resources or services with a specific time limit. Considering the features such as hierarchical view of data, quick and advanced search, and quick response, we use LDAP service to improve the load of multi-attribute data and reduce the maintenance work. According to the model in [8], we define some rules which the system decides dynamically some of the role assignments and deletions for users based on the attributes of the users. From our experience in practical environment, the extended model helps the enterprise to reduce administration costs.

### 3.2 Caching Policy

To reduce the delay experienced by the user and improve performance of WDSG, we use two kinds of caching technique: File caching and Memory Map. The file caching is simple, which saves web resources frequently requested from backend servers to make them as local services with certain time limit (a Time-To-Live-based coherency approach, TTL for short). However, we have to take many related techniques based on Java NIO to implement the memory map caching, for example, FileChannel, MappedByteBuffer, WeakHashMap and garbage collection, etc.[4]. The lifetime and caching method of all the cached objects can be flexibly configured using our own tools. In our work, currently, client validation is typically accomplished by an If-Modified-Since (IMS) HTTP Request and we have also used a standard LRU (Least-Recently-Used) cache replacement policy. In order to improve performance of system, enterprises may consider it as a edge proxy in SPREAD architecture [9]. In addition, all validated results from the access control model are cached with security policy, which greatly reduce response time of authentication and authorization.

## 4 Performance

### 4.1 Theoretical Analysis

Firstly, we examine the difference of handling concurrency between traditional blocking I/O based design and Java NIO API based design with analyzing, which is a very important issue for the performance of large-scale server. We assume that the letters  $n$ ,  $bs$ ,  $t_i$ , and  $ts$  denote the numbers of concurrent requests, the numbers of Byte bytes through network for per second, the cost time of handling the  $i$ -th (for example, ninetieth) request in program ( $z_i$  denotes its Byte bytes), and the numbers of all threads handling requests of based blocking I/O server. Assuming that total request handling time is the sum of the transmitting time on the network and handling time inside procedures respectively, then we have the formulas:

Blocking I/O:

$$\left(\sum_{i=1}^n z_i\right)/(bs * \min(ts, n)) + \sum_{i=1}^n t_i \quad (1)$$

Non-blocking I/O:

$$\left(\sum_{i=1}^n z_i\right)/(bs * n) + \sum_{i=1}^n t_i \quad (2)$$

Difference of (1) and (2):

$$\left(\sum_{i=1}^n z_i/bs\right)(1/\min(ts, n) - 1/n) \quad (3)$$

Generally, we deduce the formula

$$\left(\sum_{i=1}^n z_i / bs\right)(1/ts - 1/n) \quad (4)$$

because of the numbers of threads handling requests of large-scale server less than that of concurrent requests. If the sizes of every requested resources are same to each other, then we have the very simple formula,

$$(z_i / bs)(n/ts - 1) \quad (5)$$

It shows that if more requests are made at the same time, non-blocking I/O based server is superior to traditional blocking I/O based server. In addition, experimental data about the comparison of the performance between non-blocking I/O and traditional blocking I/O based HTTP servers is available in [10], here we omit these details.

## 4.2 Testing in Practical Environment

To perform the experiments, an Intel CPU 2.0 GHz machine with 512M of memory was used to run the WDSG, and two Intel CPU 2.4 GHz machines each with 512M of memory was used to emulate concurrent users' requests. In addition, two Intel CPU 2.4 GHz machines each with 512M of memory was used to provide users' information service and resource service respectively. All machines were running Windows 2000 Server with JDK1.4 attached JDK1.5 package and connected each other through a 100 Mbps Ethernet interface.

We used Apache JMeter<sup>3</sup> to emulate concurrent users' requests. The JMeter produced clients with the number of threads:1,5,10,15 and 500,1000,...4500 and each client produced 5 requests (the size of the returned page was 5KB) in a session. Each test ran for a time period of 2 minutes with caching enabled and disabled in WDSG respectively. Fig 3 shows that these curves of the average response time to get the page from a backend server with HTTP (some experiments without access control can be found in [10]) and HTTPS connection. According to Fig 3 shown, we can see that the difference of direct access to the resources and access through WDSG with caching is a very small value, and the increasing of average response time is very slow for the latter. It shows that the performance of the system is sufficient for large-scale enterprise environment. In addition, we can also see that there are obvious improvements for response time with caching.

The system has been deployed in a large-scale enterprise<sup>4</sup> and has exhibited sound and stable performances for two years. The approximate number of users is 8000. There are several backend servers based on multi-platform in the enterprise environment, for example, custom service management system (CSM), supply

<sup>3</sup> Apache JMeter, <http://jakarta.apache.org/jmeter/>

<sup>4</sup> TCL corporation <http://www.tcl.com>

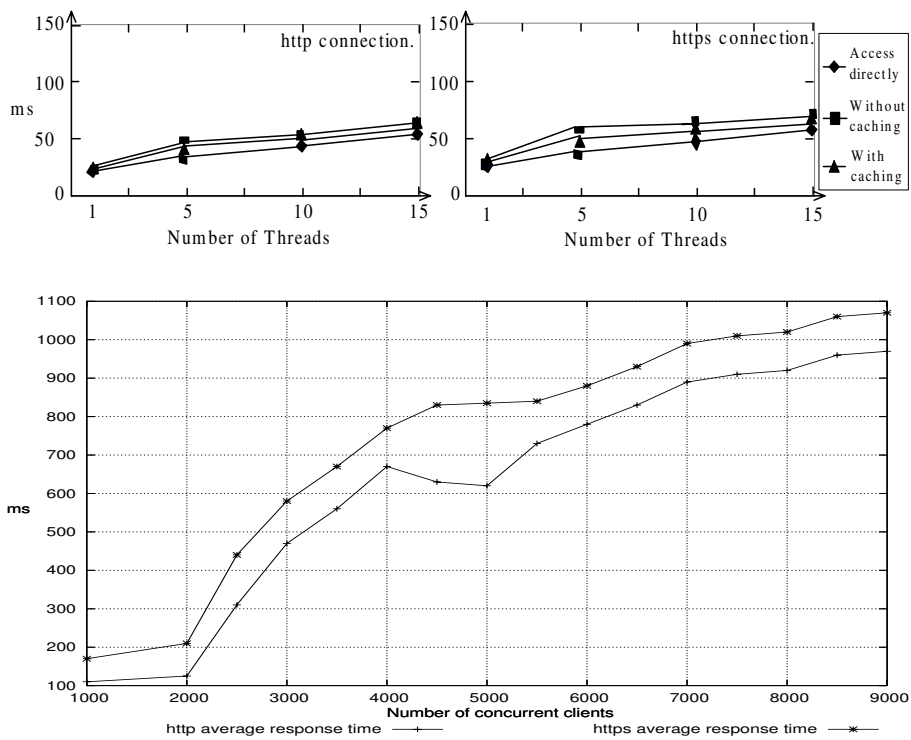


Fig. 3. Curves of the average response time using http and https connection

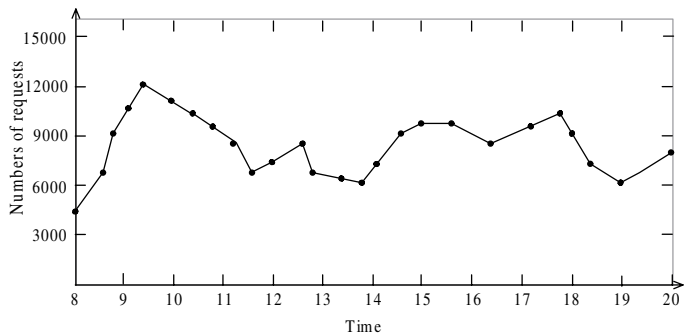


Fig. 4. Curves of the numbers of the requests in a work day

chain management system (SCM), document management system (DMS). After analyzing several days' log files randomly, we get the numbers of the requests through the system in a work day (from 8 o'clock in the morning to 8 o'clock in the evening). Fig 4 shows that curves of the numbers of the requests in a work day.



## 5 Conclusions and Future Work

In this paper, we present the design and implementation of WDSG with the techniques based on java NIO, which helps a system support massive concurrency and avoid the pitfalls which arise with traditional blocking I/O based design.

Considering the networking aspects such as accepting sockets, reading and writing the data from and to the network, we use the new traits of the Java NIO available in JDK1.4 to implement WDSG. Also, we implement the Web security access control server with the techniques of event-driving, secure cookies, SSL and caches based on the framework of PKI. WDSG provides the Single Sign-On to multiple Web applications. We also implement the delegation (distribution, dispatch) of the administration privileges of Web resources to a special module that helps us lower administration costs through Administrative RBAC. The system has been deployed in a large-scale enterprise and has exhibited sound and stable performances for two years.

Our next step is to improve our access control policy with NIST-RBAC Standard [2] and ANSI-RBAC [11]. In addition, we will observe related techniques and specifications in order to improve performance of the very system continuously.

## References

1. Zhao, C., Chen, Y., Xu, D., Nuermainaiti, Heilili, Lin, Z.: Integrative security management for web-based enterprise applications. In: WAIM 2005, LNCS 3739. (2005) 618–625
2. Ferraiolo, D., Sandhu, R., Gavrila, S., Kuhn, D., Chandramoli: Proposed nist standard for role-based access control. In: ACM Transactions on Information and System Security (TISSEC). Volume 4. (2001) 224–274
3. Park, J., Sandhu, R.: Secure cookies on the web. In: IEEE Internet Computing. Volume 4. (2000) 36–45
4. SUN New I/O APIs. <http://java.sun.com/j2se/1.4.2/docs/guide/nio/>.
5. Osborn, S., Sandhu, R., Munawer, Q.: Configuring role-based access control to enforce mandatory and discretionary access control policies. In: ACM Transactions on Information and System Security. Volume 3. (2000) 85–106
6. R.Housley, SPYRUS, W.Ford, VeriSign, W.Polk, NIST, D.Solo, Citicorp: Internet x.509 public key infrastructure certificate and crl profile, network working group request for comments: 2459 category: Standards track (1999) <http://www.ietf.org/rfc/rfc2459.txt>.
7. Sandhu, R., Coyne, E., Feinstein, H., Youman, C.: Role-based access control models. In: IEEE Computer. Volume 29. (1996) 38–47
8. Al-Kahtani, M.A., Sandhu, R.: A model for attribute-based user-role assignment. In: In Proceedings of the 18th Annual Computer Security Applications Conference, Las Vegas, Nevada, USA (2002) 353–362
9. Rodriguez, P., Sibai, S.: Spread: scalable platform for reliable and efficient automated distribution. In: The International Journal of Computer and Telecommunications Networking. Volume 33. (2000) 33–49

10. Beltran, V., Carrera, D., Torres, J., Ayguade, E.: Evaluating the scalability of java event-driven web server. In: Intelligent Computer Communication and Processing. Volume 0. (2004) 134–142
11. American National Standards Institute, Inc. Role-based access control. ANSI INCITS 359-2004. <http://csrc.nist.gov/rbac/>.

# Microaggregation for Database and Location Privacy

Josep Domingo-Ferrer

Rovira i Virgili University of Tarragona  
Dept. of Computer Engineering and Maths  
Av. Països Catalans 26  
E-43007 Tarragona, Catalonia  
josep.domingo@urv.cat

**Abstract.** Data aggregation is a central principle underlying many applications in computer science, from artificial intelligence to data security and privacy. Microaggregation is a special clustering problem where the goal is to cluster a set of points into groups of at least  $k$  points in such a way that groups are as homogeneous as possible. A usual homogeneity criterion is the minimization of the within-groups sum of squares. Microaggregation appeared in connection with anonymization of statistical databases. When discussing microaggregation for information systems, points are database records. This paper extends the use of microaggregation for  $k$ -anonymity to implement the recent property of  $p$ -sensitive  $k$ -anonymity in a more unified and less disruptive way. Then location privacy is investigated: two enhanced protocols based on a trusted-third party (TTP) are proposed and thereafter microaggregation is used to design a new TTP-free protocol for location privacy.

**Keywords:** Microaggregation,  $k$ -Anonymity, Statistical database privacy, Location privacy, Wireless systems.

## 1 Introduction

Microaggregation [3, 4] is a special clustering problem which became relevant in connection with privacy in statistical databases, a discipline also known as statistical disclosure control (SDC) whose purpose is to prevent released individual data records (microdata) from being linkable with the respondents they correspond to. The microaggregation problem is to cluster a set of  $d$ -dimensional points (points represent records in the database application) into groups of at least  $k$  points in such a way that groups are as homogeneous as possible.

Let  $\mathbf{X}$  be a dataset formed by  $n$  points in a  $d$ -dimensional numerical space. Microaggregation is operationally defined in terms of two steps. Given a parameter  $k$ , the first step partitions points in  $\mathbf{X}$  into groups of at least  $k$  points each. The second step replaces each point by the centroid of its group to obtain the masked dataset  $\mathbf{X}'$ . In a microaggregated dataset where points correspond to individuals (*e.g.* each point is a record with individual data, and the attribute values in the record are the point co-ordinates), no re-identification within a

group is possible, because all  $k$  points in a group are identical: the best that an intruder can hope is to track the group where a target individual has been masked into.

Microaggregating with minimum information loss has been known to be an important—and difficult—issue ever since microaggregation was invented as an SDC masking method for microdata. However, it was often argued that optimality in SDC is not just about minimum information loss but about the best tradeoff between low information loss and low disclosure risk. The recent application [8] of microaggregation to achieve  $k$ -anonymity [24, 23, 26, 27] for numerical microdata leaves no excuse to circumvent the problem of trying to reduce information loss as much as possible: once a value  $k$  is selected that keeps the re-identification risk low enough, the only job left is to  $k$ -anonymize (that is, to microaggregate) with as little information loss as possible.

A partition  $P$  such that all of its groups have size at least  $k$  is called a  $k$ -partition ([4]) and microaggregation with parameter  $k$  is sometimes denoted as  $k$ -microaggregation.

In [4], optimal microaggregation is defined as the one yielding a  $k$ -partition maximizing the within-groups homogeneity. The rationale is that, the more homogeneous the points in a group, the less variability reduction when replacing those points by their centroid and thus the less information loss. The within-groups sum of squares  $SSE$  is a usual measure of within-groups homogeneity in clustering [34, 9, 11, 12], so a reasonable optimality criterion for a  $k$ -partition  $P = \{G_1, \dots, G_g\}$  is to minimize  $SSE$ , *i.e.* to minimize

$$SSE(P) = \sum_{i=1}^g \sum_{j=1}^{|G_i|} (x_{ij} - c(G_i))'(x_{ij} - c(G_i))$$

where  $|G_i|$  is the number of points in the  $i$ -th group,  $c(G_i)$  is the centroid of the  $i$ -th group and  $x_{ij}$  is the  $j$ -th point in the  $i$ -th group. It was shown in [4] that groups in the optimal  $k$ -partition have sizes between  $k$  and  $2k - 1$ .

For the univariate case (one-dimensional points), the optimal  $k$ -partition can be computed in polynomial time using the algorithm in [13]. For the multivariate case ( $d > 1$ ), the optimal microaggregation problem has been shown to be NP-hard ([20]). Therefore, algorithms for multivariate microaggregation are heuristic [2, 4, 25, 14, 15]. Quite recently, the first approximation heuristic for multivariate microaggregation has been presented ([6]); it yields a  $k$ -partition whose  $SSE$  is no more than a certain multiple of the minimum  $SSE$ .

Microaggregation can be extended for categorical spaces (that is, for records with categorical attributes) if the Euclidean distance is replaced by an appropriate categorical distance and the mean is replaced by a suitable categorical average when computing centroids [30].

## 1.1 Contribution and Plan of This Paper

In this work we will review the use of microaggregation to provide disclosure control and  $k$ -anonymity in statistical databases. In this context we will give

new results on how to adapt microaggregation to implement the recently defined property of  $p$ -sensitive  $k$ -anonymity in a more unified and less disruptive way (without generalizations nor suppressions). We will then investigate how microaggregation can be adapted for application to location privacy.

Section 2 deals with the application to statistical databases. Section 3 is about the new application to location privacy. Conclusions and lines for future research are sketched in Section 4.

## 2 Microaggregation for Statistical Databases

Microaggregation became known as a statistical disclosure control technique proposed by Eurostat researchers in the early nineties [3]. The heuristics developed at that time split the records in a dataset into groups of exactly  $k$  records, except for the last group, which contained between  $k$  and  $2k - 1$  records. The operation of such heuristics was basically *univariate*: either the records were projected onto a single dimension —*e.g.* using the first principal component, the sum of  $z$ -scores or a particular attribute— or microaggregation was conducted independently for each attribute in the dataset —what is known as individual ranking microaggregation [21]—.

Projecting multivariate data onto a single dimension caused a very high loss of information (data utility) [7], while using individual ranking provides very little protection against disclosure risk [7, 5]. The authors of [4] were the first to define *multivariate* microaggregation as a cardinality-constrained multivariate clustering problem (in the way described in Section 1) and give a heuristic for it. Going multivariate allowed a tradeoff between information loss and disclosure risk to be struck and turned microaggregation into one of the best SDC methods for microdata [7].

Since it was proposed at Eurostat, microaggregation has been used in Germany [22, 16], Italy [21] and several other countries, detailed in the surveys [32, 33] by the United Nations Economic Commission for Europe.

### 2.1 $k$ -Anonymity Using Microaggregation

In [8], microaggregation was shown to be useful to implement the property of  $k$ -anonymity [24, 23, 26, 27]. To recall the definition of  $k$ -anonymity, we need to enumerate the various (non-disjoint) types of attributes that can appear in a microdata set  $\mathbf{X}$ :

- *Identifiers*. These are attributes that *unambiguously* identify the respondent. Examples are passport number, social security number, full name, etc. Since our objective is to prevent confidential information from being linked to specific respondents, we will assume in what follows that, in a pre-processing step, identifiers in  $\mathbf{X}$  have been removed/encrypted.
- *Key attributes*. Borrowing the definition from [1, 23], key attributes are those in  $\mathbf{X}$  that, in combination, can be linked with external information to re-identify (some of) the respondents to whom (some of) the records in  $\mathbf{X}$

refer. Unlike identifiers, key attributes cannot be removed from  $\mathbf{X}$ , because any attribute is potentially a key attribute.

- *Confidential outcome attributes.* These are attributes which contain sensitive information on the respondent. Examples are salary, religion, political affiliation, health condition, etc.

Now, the  $k$ -anonymity property can be stated as:

**Definition 1 ( $k$ -Anonymity).** *A dataset is said to satisfy  $k$ -anonymity for  $k > 1$  if, for each combination of values of key attributes (e.g. name, address, age, gender, etc.), at least  $k$  records exist in the dataset sharing that combination.*

In the seminal  $k$ -anonymity papers [24, 23, 26, 27], the computational procedure suggested to  $k$ -anonymize a dataset relies on suppressions and generalizations. The drawbacks of partially suppressed and coarsened data for analysis were highlighted in [8]. Joint multivariate microaggregation of all key attributes with minimum group size  $k$  was proposed in [8] as an alternative to achieve  $k$ -anonymity; besides being simpler, this alternative has the advantage of yielding complete data without any coarsening (nor categorization in the case of numerical data).

## 2.2 A New Property: $p$ -Sensitive $k$ -Anonymity Using Microaggregation

We will show here how microaggregation can be used to implement a recently defined privacy property called  $p$ -sensitive  $k$ -anonymity [31]. This is an evolution of  $k$ -anonymity whose motivation is to avoid that records sharing a combination of key attributes in a  $k$ -anonymous data set also share the values for one or more confidential attributes. In this case,  $k$ -anonymity does not offer enough protection.

*Example 1.* Imagine that an individual's health record is  $k$ -anonymized into a group of  $k$  patients with  $k$ -anonymized key attributes values  $Age = "30"$ ,  $Height = "180\text{ cm}"$  and  $Weight = "80\text{ kg}"$ . Now, if all  $k$  patients share the confidential attribute value  $Disease = "AIDS"$ ,  $k$ -anonymization is useless, because an intruder who uses the key attributes ( $Age$ ,  $Height$ ,  $Weight$ ) can link an external identified record

*(Name="John Smith", Age="31", Height="179", Weight="81")*

with the above group of  $k$  patients and infer that John Smith suffers from AIDS. □

Based on the above, the  $p$ -sensitive  $k$ -anonymity property is defined as:

**Definition 2 ( $p$ -Sensitive  $k$ -anonymity).** *A dataset is said to satisfy  $p$ -sensitive  $k$ -anonymity for  $k > 1$  and  $p \leq k$  if it satisfies  $k$ -anonymity and, for each group of tuples with the same combination of key attribute values that exists in the dataset, the number of distinct values for each confidential attribute is at least  $p$  within the same group.*

In the above example,  $p$ -sensitive  $k$ -anonymity would require in particular that there be at least  $p$  different diseases in each group of people sharing the same  $k$ -anonymized age, height and weight.

The computational approach proposed in [31] to achieve  $p$ -sensitive  $k$ -anonymity is an extension of the generalization/suppression procedure proposed in papers [24, 23, 26, 27]. Therefore, it shares the same shortcomings pointed out in [8]:  $p$ -sensitive  $k$ -anonymized data are a coarsened and partially suppressed version of original data. Generalization is undesirable for a number of reasons: i) it transforms numerical (continuous) attributes into categorical ones; ii) if applied to all records, it causes a great loss of information and, if applied only locally, it introduces new categories that co-exist with the old ones and complicate subsequent analyses. On the other hand, partially suppressed data cannot be analyzed using standard statistical tools (techniques for censored data are needed).

Like we did for  $k$ -anonymity in [8], we propose here to achieve  $p$ -sensitive  $k$ -anonymity via microaggregation. The proposed algorithm is as follows.

**Algorithm 1** ( *$p$ -sensitive  $k$ -anonymization*). ( $\mathbf{X}$ : dataset,  $k, p$ : integers)

1. If  $p > k$ , signal an error (" $p$ -sensitive  $k$ -anonymity infeasible") and exit the Algorithm.
2. If the number of distinct values for any confidential attribute in  $\mathbf{X}$  is less than  $p$  over the entire dataset, signal an error (" $p$ -sensitive  $k$ -anonymity infeasible") and exit the Algorithm.
3.  $k$ -Anonymize  $\mathbf{X}$  using microaggregation as described in Section 2.1. Let  $\mathbf{X}'$  be the microaggregated,  $k$ -anonymized dataset.
4. Let  $\hat{k} := k$ .
5. While  $p$ -sensitive  $k$ -anonymity does not hold for  $\mathbf{X}'$  do:
  - (a) Let  $\hat{k} := \hat{k} + 1$ .
  - (b)  $\hat{k}$ -Anonymize  $\mathbf{X}$  using microaggregation. Let  $\mathbf{X}'$  be the  $\hat{k}$ -anonymized dataset.

The above algorithm is based on the following facts:

- A  $k + 1$ -anonymous dataset is also  $k$ -anonymous.
- By increasing the minimum group size, the number of distinct values for confidential attributes within a group also increases.

The resulting  $p$ -sensitive  $k$ -anonymous dataset does not contain coarsened or partially suppressed data. This makes its analysis and exploitation easier, with the additional advantage that numerical continuous attributes are not categorized.

### 3 Microaggregation for Location Privacy

The growth of location-detection devices (*e.g.* cellular phones, GPS-like devices, RFIDs and handheld devices) along with wireless communications has fostered

the development of location-based commercial services [17, 18, 28, 29]. These are applications that deliver specific information to their users *based* on their current location (*e.g.* list of hotels or pharmacies near the place where you are, etc.).

As noted in [35], although location-based services may be very convenient, they threaten the privacy and security of their customers. Several approaches to location privacy have been proposed, an overview of which is given in [19]. The latter paper also proposes a model in which a *location anonymizer* acts as a third party between the mobile users and the location-based database servers. The interaction between user, anonymizer and database server is completely mediated by the anonymizer according to the following protocol:

### Protocol 1 (Mediator with location anonymization)

1. User  $U$  sends her query to the anonymizer  $A$ .
2.  $A$  anonymizes the location of  $U$  and forwards the query to the database server  $DS$ .
3.  $DS$  returns the query answer to  $A$ .
4.  $A$  forwards the query answer to  $U$ .

The location anonymizer's aim is to protect the privacy of queries issued by roaming users who sent their location as an input parameter for the query. The approach in [19] is for the location anonymizer to provide a special form of location  $k$ -anonymity. Location privacy differs from the database privacy problem dealt with in Section 2 in some respects:

- Locations are updated more frequently than typical databases. Therefore,  $k$ -anonymity algorithms for location privacy must be faster than those for databases.
- $k$ -Anonymity alone may not be enough. If  $k$  roaming users happen to be in nearly the same place (*e.g.* a department store) and one of them issues a query, no privacy is gained if the location anonymizer forwards to the database server the minimum bounding rectangle containing the locations of the  $k$  users instead of the querying user's locations.

In view of the above peculiarities, the following requirements are stated in [19] on the algorithm used by the location anonymizer to *cloak* a user's location into a spatial region containing it:

1. The *cloaking* region should contain at least  $k$  users to provide  $k$ -anonymity. In addition to this, the cloaking region should cover an area  $A$  such that  $A_{min} < A < A_{max}$ . The minimum bound  $A_{min}$  is a privacy parameter and the maximum bound  $A_{max}$  is a utility parameter (too large a cloaking region is useless to obtain location-based services from the database server).
2. An intruder should not be able to know the exact location of the user within the cloaking region.
3. The cloaking algorithm should be computationally efficient to cope in real time with the continuous movement of mobile users.



*Note 1.* As a side remark, it is interesting to note that the  $k$ -anonymity used for location privacy in the literature [19, 10] is somewhat different from the standard  $k$ -anonymity discussed in Section 2. *What is perturbed (cloaked) in location privacy is the user's confidential attributes (latitude and longitude)*, whereas in database  $k$ -anonymity perturbation affects the key attributes.

### 3.1 Mediator Without Location Anonymization

In the architecture proposed in [19], a trusted third party is mediating all user queries to the database server. We claim that, in this context, there is no need for anonymizing the user location, since the mediator can withhold all the user's identifiers and key attributes. This would result in the following protocol:

#### Protocol 2 (Mediator without location anonymization)

1. User  $U$  sends her query to the mediator  $M$ .
2.  $M$  suppresses any identifiers or key attributes in the query and forwards only the question in the query plus the exact location of  $U$  (latitude, longitude) to the database server  $DS$ .
3.  $DS$  returns the query answer to  $M$ , based on the location provided.
4.  $M$  forwards the query answer to  $U$ .

Note that the  $DS$  only receives questions and (latitude, longitude) pairs without any identifying information. Therefore  $DS$  is unable to link those locations with the users they correspond to. Additionally, since exact locations are sent to the database server, query answers can be more accurate and useful. *E.g.* if the query is about hotels or pharmacies near the user's location, only those really close hotels or pharmacies will be included in the answer.

### 3.2 Location Anonymization Without Mediation

Only when there is direct interaction between the user and the database server is location anonymization really needed. This is the situation described by the following protocol:

#### Protocol 3 (Location anonymization without mediation)

1. User  $U$  requests the anonymizer  $A$  to anonymize her location.
2.  $A$  anonymizes the location of  $U$ . Like in Protocol 1, this is done by cloaking it with the locations of at least  $k - 1$  additional users, while taking spatial constraints into account (see above).  $A$  returns the anonymized location to  $U$ .
3.  $U$  sends her query to  $DS$ , using her anonymized location instead of her real location.
4.  $DS$  returns the query answer to  $U$ , based on the location provided.

Protocol 3 is a clear improvement regarding user privacy, because the anonymizer does not receive either user queries or their answers (unlike the anonymizer in Protocol 1 or the mediator in Protocol 2).

### 3.3 TTP-Free Location Anonymization Through Microaggregation

One might argue against Protocols 1, 2 and 3 that their use of a trusted third party (TTP) is a privacy weakness. Indeed, the TTP (anonymizer or mediator) learns the whereabouts and the identities of all users. Additionally, in Protocols 1 and 2, the TTP learns the content of the queries and the answers.

We present next a TTP-free protocol, whereby users do not need to disclose their exact location or their queries. The operating principle is to compute the centroid of at least  $k$  perturbed user locations and feed it directly to the database server.

#### Protocol 4 (TTP-free location anonymization)

1. User  $U$  does:
  - (a) Generate a perturbed version of her location  $(x, y)$  by adding Gaussian noise to her latitude  $x$  and longitude  $y$ , that is

$$(x', y') = (x + \epsilon_x, y + \epsilon_y)$$

where  $\epsilon_x$  is drawn from a  $N(0, \sigma_x^2)$  random variable and  $\epsilon_y$  is drawn from a  $N(0, \sigma_y^2)$  random variable. All users use the same  $\sigma_x, \sigma_y$ .

- (b) Broadcast a message containing  $(x', y')$  to request neighbors to return perturbed versions of their locations.
  - (c) Among the replies received, select  $k - 1$  neighbors such that the group  $G$  formed by them plus  $(x', y')$  spans an area  $A$  satisfying  $A_{min} < A < A_{max}$ , where  $A_{min}$  and  $A_{max}$  are input parameters with the meaning defined above. If there are several feasible groups  $G$ , choose the one minimizing the within-group sum of squares  $SSE$ . If there is no feasible  $G$  (not enough neighbors have replied or the above spatial constraint cannot be met) go back to Step 1b.
  - (d) Compute the centroid  $(c'_x, c'_y)$  of  $G$ .
  - (e) Send her query to  $DS$ , using  $(c'_x, c'_y)$  as location instead of  $U$ 's real location.
2.  $DS$  returns the query answer to  $U$ , based on the location provided.

**Lemma 1.** *If each user perturbs her location using Gaussian noise  $N(0, \sigma_x^2)$  for latitude and  $N(0, \sigma_y^2)$  for longitude, the standard errors of the centroid  $(c'_x, c'_y)$  of  $k$  perturbed locations with respect to the centroid  $(c_x, c_y)$  of the real locations are  $\sigma_x/\sqrt{k}$  and  $\sigma_y/\sqrt{k}$ , respectively.*

**Proof.** By elementary sampling theory, the sample mean of  $k$  observations of a  $N(0, \sigma^2)$  random variable follows a  $N(0, \sigma^2/k)$  distribution. Thus,

$$(c'_x, c'_y) = (c_x + \epsilon_{c,x}, c_y + \epsilon_{c,y})$$

where  $\epsilon_{c,x}$  is drawn from a  $N(0, \sigma_x^2/n)$  random variable and  $\epsilon_{c,y}$  is drawn from a  $N(0, \sigma_y^2/k)$  random variable.  $\square$

So, the perturbation added by each user to her own location is mitigated by the centroid computation. Even for small  $k$  and moderate  $\sigma_x, \sigma_y$ , the centroid  $(c'_x, c'_y)$  sent to the database server can be expected to be close to the real centroid  $(c_x, c_y)$ . Therefore, the utility the query answer provided by the database server is largely unaffected by perturbation.

The reason for each user to perturb her own location is to defend her own privacy from collusions. Without perturbation,  $k-1$  users could collude and pool their real locations to derive the remaining user's location. With perturbation, the colluders will get no more than the perturbed location of the remaining user. If a non-roaming user has already sent one or more perturbed versions of her constant real location in previous instances of Step 1b of Protocol 4, she should evaluate the privacy risk incurred if she sends a new perturbed version: by Lemma 1, anyone can average those perturbed locations to get closer to the user's real location. A roaming user does not face that problem, since her real location is continuously changing.

## 4 Conclusions and Future Research

We have presented several applications of the principle of microaggregation to providing privacy in databases and location-based services. In the case of databases, we have extended the use of microaggregation for  $k$ -anonymity to implement the recent property of  $p$ -sensitive  $k$ -anonymity. In the case of location privacy, we have proposed two TTP-based protocols which improve on the state of the art. Despite the improvements, TTP-based privacy remains weak, so we have also proposed a TTP-free protocol for location privacy based on microaggregation.

Future research will include refining and tuning the parameters for using microaggregation to implement  $p$ -sensitive  $k$ -anonymity and TTP-free location privacy.

## Acknowledgments

The author is partly supported by the Catalan government under grant 2005 SGR 00446, and by the Spanish Ministry of Science and Education through project SEG2004-04352-C04-01 "PROPRIETAS".

## References

1. T. Dalenius. Finding a needle in a haystack - or identifying anonymous census records. *Journal of Official Statistics*, 2(3):329–336, 1986.
2. D. Defays and N. Anwar. Micro-aggregation: a generic method. In *Proceedings of the 2nd International Symposium on Statistical Confidentiality*, pages 69–78, Luxembourg, 1995. Eurostat.
3. D. Defays and P. Nanopoulos. Panels of enterprises and confidentiality: the small aggregates method. In *Proc. of 92 Symposium on Design and Analysis of Longitudinal Surveys*, pages 195–204, Ottawa, 1993. Statistics Canada.

4. J. Domingo-Ferrer and J. M. Mateo-Sanz. Practical data-oriented microaggregation for statistical disclosure control. *IEEE Transactions on Knowledge and Data Engineering*, 14(1):189–201, 2002.
5. J. Domingo-Ferrer, Josep M. Mateo-Sanz, A. Oganian, and À. Torres. On the security of microaggregation with individual ranking: analytical attacks. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):477–492, 2002.
6. J. Domingo-Ferrer, F. Sebé, and A. Solanas. A polynomial-time approximation to optimal multivariate microaggregation. *Manuscript*, 2005.
7. J. Domingo-Ferrer and V. Torra. A quantitative comparison of disclosure control methods for microdata. In P. Doyle, J. I. Lane, J. J. M. Theeuwes, and L. Zayatz, editors, *Confidentiality, Disclosure and Data Access: Theory and Practical Applications for Statistical Agencies*, pages 111–134, Amsterdam, 2001. North-Holland. <http://vneumann.etse.urv.es/publications/bcpi>.
8. J. Domingo-Ferrer and V. Torra. Ordinal, continuous and heterogeneous  $k$ -anonymity through microaggregation. *Data Mining and Knowledge Discovery*, 11(2):195–212, 2005.
9. A. W. F. Edwards and L. L. Cavalli-Sforza. A method for cluster analysis. *Biometrics*, 21:362–375, 1965.
10. B. Gedik and L. Liu. A customizable  $k$ -anonymity model for protecting location privacy. In *Proceedings of the International Conference on Distributed Computing Systems-ICDCS*, 2005.
11. A. D. Gordon and J. T. Henderson. An algorithm for euclidean sum of squares classification. *Biometrics*, 33:355–362, 1977.
12. P. Hansen, B. Jaumard, and N. Mladenovic. Minimum sum of squares clustering in a low dimensional space. *Journal of Classification*, 15:37–55, 1998.
13. S. L. Hansen and S. Mukherjee. A polynomial algorithm for optimal univariate microaggregation. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):1043–1044, 2003.
14. A. Hundepool, A. Van de Wetering, R. Ramaswamy, L. Franconi, A. Capobianchi, P.-P. DeWolf, J. Domingo-Ferrer, V. Torra, R. Brand, and S. Giessing.  *$\mu$ -ARGUS version 4.0 Software and User's Manual*. Statistics Netherlands, Voorburg NL, may 2005. <http://neon.vb.cbs.nl/casc>.
15. M. Laszlo and S. Mukherjee. Minimum spanning tree partitioning algorithm for microaggregation. *IEEE Transactions on Knowledge and Data Engineering*, 17(7):902–911, 2005.
16. R. Lenz and D. Vorgrimler. Matching german turnover tax statistics. Technical Report FDZ-Arbeitspapier Nr. 4, Statistische Ämter des Bundes und der Länder-Forschungsdatenzentren, 2005.
17. <http://www.locatrix.com>.
18. <http://www.mapinfo.com>.
19. M. F. Mokbel. Towards privacy-aware location-based database servers. In *Proceedings of the Second International Workshop on Privacy Data Management-PDM'2006*, Atlanta, GA, 2006. IEEE Computer Society.
20. A. Oganian and J. Domingo-Ferrer. On the complexity of optimal microaggregation for statistical disclosure control. *Statistical Journal of the United Nations Economic Commission for Europe*, 18(4):345–354, 2001.
21. D. Pagliuca and G. Seri. Some results of individual ranking method on the system of enterprise accounts annual survey, 1999. Esprit SDC Project, Deliverable MI-3/D2.

22. M. Rosemann. Erste Ergebnisse von vergleichenden Untersuchungen mit anonymisierten und nicht anonymisierten Einzeldaten am Beispiel der Kostenstrukturerhebung und der Umsatzsteuerstatistik. In G. Ronning and R. Gnoss, editors, *Anonymisierung wirtschaftsstatistischer Einzeldaten*, pages 154–183, Wiesbaden, Germany, 2003. Statistisches Bundesamt.
23. P. Samarati. Protecting respondents' identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027, 2001.
24. P. Samarati and L. Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, SRI International, 1998.
25. G. Sande. Exact and approximate methods for data directed microaggregation in one or more dimensions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):459–476, 2002.
26. L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal of Uncertainty, Fuzziness and Knowledge Based Systems*, 10(5):571–588, 2002.
27. L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge Based Systems*, 10(5):557–570, 2002.
28. <http://www.targusinfo.com>.
29. <http://www.telostar.com>.
30. V. Torra. Microaggregation for categorical variables: a median based approach. In *Lecture Notes in Computer Science*, vol. 3050, pp. 162–174, Berlin Heidelberg, 2004. Springer.
31. T. M. Truta and B. Vinay. Privacy protection:  $p$ -sensitive  $k$ -anonymity property. *Manuscript*, 2005.
32. UNECE. United nations economic commission for europe. questionnaire on disclosure and confidentiality - summary of replies. In *1st Joint UNECE/Eurostat Work Session on Statistical Data Confidentiality*, Thessaloniki, Greece, 1999.
33. UNECE. United nations economic commission for europe. questionnaire on disclosure and confidentiality - summary of replies. In *2nd Joint UNECE/Eurostat Work Session on Statistical Data Confidentiality*, Skopje, Macedonia, 2001.
34. J. H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58:236–244, 1963.
35. J. Warrior, E. McHenry, and K. McGee. They know where you are. *IEEE Spectrum*, 40(7):20–25, 2003.

# Efficient Caching Strategies for Gnutella-Like Systems to Achieve Anonymity in Unstructured P2P File Sharing

Byung Ryong Kim<sup>1</sup> and Ki Chang Kim<sup>2</sup>

<sup>1</sup> School of Computer and Science Engineering, Inha Univ., 253, YongHyun-Dong,  
Nam-Ku, Incheon, 402-751, Korea  
dooly@inha.ac.kr

<sup>2</sup> School of Information and Communication Engineering, Inha Univ., 253, YongHyun-Dong,  
Nam-Ku, Incheon, 402-751, Korea  
kchang@inha.ac.kr

**Abstract.** A critical problem in a Peer-to-Peer file sharing system is how to protect the anonymity of agent nodes when providing efficient data access services. Most of existing technique mostly focus on how to provide the initiator anonymity and privacy, but leave the anonymity of the server. Our technique is similar to Onion routing in that it uses a proxy node to hide the identity of the client and the server. But it differs from it in that the proxy node is selected dynamically for each session and not fixed as in Onion routing. The fixed proxy might be convenient to use and is useful when loaded with additional features such as encryption as in Onion routing. However it is a sort of compromise from P2P point of view: we are introducing something similar to a control center that could become a bottleneck in the network. Temporary proxy doesn't have this problem. And our technique provides caching of a retrieved data item on the reverse path back to the requestor. Caching in our paper is primarily for providing publisher anonymity and security. In this paper, we show a technique to find such proxy and caching, explain the rationale behind the decision, and prove its effectiveness through experimentation.

## 1 Introduction

Recently lots of attention is given to a distributed framework called Peer-to-Peer (P2P) network. A P2P network consists of a large number of peers interconnected together to share all kinds of digital content. In a P2P network, there is no central or dedicated server and all peers are independent from each other. P2P is characterized by resource sharing among computing machines that participate in the network. In Server/Client environment, the user sends query messages to the server, and a popular server suffers a heavy load resulting degraded service. P2P network alleviates this problem by distributing the server data among the participating nodes and providing a mechanism to locate the data dynamically. This way copies of data are scattered around in the network, and the load concentration problem can be handled; however, it became harder and more expensive to find out which nodes have the data.

Currently locating data is done via Flooding techniques as in FreeNet[1] or Gnutella[2] or Distributed Hash Table techniques as in Tapestry[3,4], Can[5], or

Chord[6]. Flooding model involves a repeated broadcasting of a query and inevitably causes a heavy traffic. Distributed Hash Table reduces query traffic considerably but still has the problem of keeping the table fresh. Both, however, do not provide anonymity for server locations and could expose servers to DoS or storage flooding attack or anonymity-breaking attacks[7,8,9,10].

Our technique is similar to Onion routing[4] in that it uses a proxy node to hide the identity of the client and the server. But it differs from it in that the proxy node is selected dynamically for each session and not fixed as in Onion routing. The fixed proxy might be convenient to use and is useful when loaded with additional features such as encryption as in Onion routing. However it is a sort of compromise from P2P point of view: we are introducing something similar to a control center that could become a bottleneck in the network. Temporary proxy doesn't have this problem: it is used once and discarded, and later another node is selected as the proxy. However using temporary proxy has a different problem: it is hard to find the right proxy. It should be in a particular path set up between the client and the server, and it should be in some position where the proxying service can be most efficiently performed. In this paper, we show a technique to find such proxy, explain the rationale behind the decision, and prove its effectiveness through experimentation.

The rest of the paper is as follows. Section 2 summarizes previous researches on providing anonymity in P2P network. Section 3 looks at the probabilistic characteristics in P2P packets and caching technique which will be used in our algorithm that is explained in Section 4. Section 5 shows the experimental results, and Section 6 concludes the paper.

## 2 Anonymous Systems

Anonymity problem in P2P networks is studied in several strands of related work. Anonymity for file access is originally proposed in IP network. Since servers in the traditional Client-Server computing model are always overt and fixed, previous work on anonymity mainly focused on the provision of initiator anonymity for clients. Anonymizer[11] and LPWA [12] introduced a proxy between clients and servers to generate consistent untraceable aliases for clients. However, this approach relies too much on the intermediate proxy and cannot fully protect requestor privacy against the logs kept by the proxy. To improve that, Onion Routing and Crowds[13] used covert paths to achieve initiator anonymity. A peer that wishes to hide its identity organizes a set of network nodes into a “covert path” with its communication partner residing at the end of the path. By changing the nodes selected for the path and varying its length with different frequency, anonymity degradation can be avoided. However, organizing and periodically reforming the covert path introduce much resource consumption and access delay. To enhance system security, Web Mixes [14] employed mixing technique to thwart timing analysis. Such mixing techniques guarantee anonymity via message padding and reordering. Although they can efficiently resist timing attacks, these techniques still provide no responder anonymity. Hordes [15] employed multiple proxies together with IP multicast technology to achieve anonymity for Internet communication. But its mechanism for group member selection is inefficient.

The primary goal for Freenet security is protecting the anonymity of requestors and inserters of files. As Freenet communication is not directed towards specific receivers, receiver anonymity is more accurately viewed as key anonymity, that is, hiding the key which is being requested or inserted. Anonymous point-to-point channels based on Chaum's mix-net scheme[16] have been implemented for email by the Mixmaster remailer[17] and for general TCP/IP traffic by onion routing and freedom[18]. Such channels are not in themselves easily suited to one-to-many publication, however, and are best viewed as a complement to Freenet since they do not provide file access and storage. Anonymity for consumers of information in the web context is provided by browser proxy services such as the Anonymizer, although they provide no protection for producers of information and do not protect consumers against logs kept by the services themselves. Private information retrieval schemes[19] provide much stronger guarantees for information consumers, but only to the extent of hiding which piece of information was retrieved from a particular server. In many cases, the fact of contacting a particular server in itself can reveal much about the information retrieved, which can only be counteracted by having every server hold all information. Reiter and Rubin's Crowds system uses a similar method of proxying requests for consumers, although Crowds does not itself store information and does not protect information producers. Berthold *et al.* propose Web MIXes, a stronger system that uses message padding and reordering and dummy messages to increase security, but again does not protect information producers.

The Rewebber[20] provides a measure of anonymity for producers of web information by means of an encrypted URL service that is essentially the inverse of an anonymizing browser proxy, but has the same difficulty of providing no protection against the operator of the service itself. Publius[21] enhances availability by distributing files as redundant shares among  $n$  web servers, only  $k$  of which are needed to reconstruct a file; however, since the identity of the servers themselves is not anonymized, an attacker might remove information by forcing the closure of  $n-k+1$  servers. The Eternity proposal[22] seeks to archive information permanently and anonymously, although it lacks specifics on how to efficiently locate stored files, making it more akin to an anonymous backup service. Free Haven[23] is an interesting anonymous publication system that uses a trust network and file trading mechanism to provide greater server accountability while maintaining anonymity.

MUTE[24] forces all intermediate nodes along the path between the client and the server node to work as proxies to protect the identities of the client and the server. Every node in the path including the client and the server thinks its previous node is the client and its next one the server. Therefore the data from the true server will be relayed node by node along the path causing a heavy traffic, especially for large multimedia files. Tarzan[25] is a peer-to-peer anonymous IP network overlay. so it works with any internet application. Its peer-to-peer design makes it decentralized, scalable, and easy to manage. But Tarzan provides anonymity to either clients or servers. Mantis[26] is similar to Crowds in that there are helping nodes to propagate the request to the candidate servers anonymously. Since Mantis preserves the anonymity of the server only, the server knows where is the client. The server sends the requested data to the client directly but in UDP hiding its IP. UDP is convenient to hide the server's identity but due to the packet loss inevitable in UDP Mantis needs additional packet retransmission mechanism. Napster[28], the first P2P application, provides no

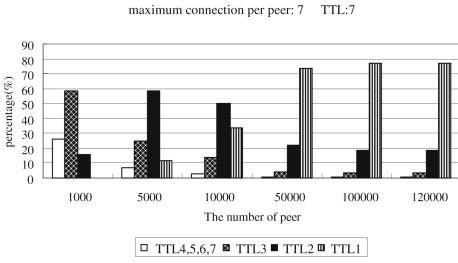


anonymity for both communication sides due to the identity information of each participant is revealed by the centralized servers. Gnutella uses flooding to search the target file and hops the response back to the requestor along the original forwarding path. Since the searching path is dynamically determined, it is rather difficult for the third to trace the route of each packet in the network. Despite the mutual anonymity achieved in the search phase, this merit is unfortunately lost later because the actual file transfer procedure is performed via direct HTTP connections between the two peers. Freenet is another protocol for pure P2P systems. It uses a depth-first algorithm to organize a forwarding path based on the dynamically updated routing information. Once the target file is located, it is delivered back to the initiator along the path and is cached on each intermediate node. All these nodes can claim themselves to be the initiator of the request or the owner of the file. Thus, Freenet is capable of providing mutual anonymity. However, the caching technique and the transfer technique consume many resources and inevitably lead to low efficiency. In addition, there exists the single point failure problem in file transfer phase. APFS [29] is P2P file sharing protocol which also employed IP multicast technology. But we are different from it in the usage of IP multicast. IP multicast is adopted here for session management in APFS, while not for file transmission. Mutual anonymity in APFS is achieved through on Onion Routing. If a certain structure, such as ring, tree or star, is imposed on P2P systems, a more efficient file locating mechanism, DHTs (Distributed Hash Tables) can be used [3,5,6]. However, most DHTs algorithms focus on finding a routing path to the destination with minimal hops and take the assumption that files are determinately distributed over the system according to some hash functions. And it doesn't consider the anonymity issue of our researches.

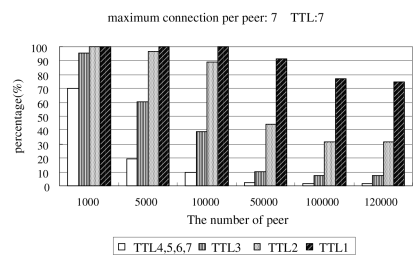
### 3 Message Distribution and Caching Techniques

In our proposed technique, most of messages are QueryHit messages, which are Query and retrieval results to retrieval data shard with Ping/Pong message periodically broadcasted for exchange and collection of connection information. The higher the number of peers participating in Gnutella network gets, the more rapidly the amount of these messages increases. Fig.1 and 2 shows that the result of broadcasting one packet once for one host and based on Newtella Gnutella Clone[30] on Gnutella protocol basis. Fig. 1 shows that the greater the number of peers increases, the greater the proportion of messages with low TTL value gets. The figure indicates that where the number of peers is big, the proportion of messages with low TTL value exceeds 50 percent and where the number of peers is big, the proportion of TTL value of 6,5,4 among the whole Ping messages is extremely little.

Fig. 2 shows the proportion of discarded Ping messages by GUID value for each TTL value. In Fig. 1 it is important when peers are considerably many although Ping messages with TTL value, 1 are virtually relayed and transferred, it has more than 75 percent of probability to ignore this Ping messages at peers that receive Ping messages. In addition when the number of peers is considerably 10,000 the proportion of messages with TTL value, 1 is more than 33% and more than 99% of those are discarded whereas the proportion of messages with TTL value, 2 is about 50 percent and



**Fig. 1.** Ping message distribution by number of peers and TTL value



**Fig. 2.** Invalid message proportion to messages of each TTL value

more than 88% of those are discarded. Ultimately TTL value, 2 holds the most valid messages since TTL value, 7,6,5,4,3 is about 16% and 50% of that is discarded. Query message also has the similar distribution with Ping message. Therefore in this study we apply zipf's Law[31,32] to message with TTL value, 2.

In order to retrieve and obtain desired information in Gnutella network, it is more beneficial to get as much connection information as possible so many Ping/ Retrieval Query messages are broadcasted through network. At this time peers receive the information directly from many peers closely connected on the basis of them and TTL value, and receive relayed retrieval Query messages or retrieval result value to relayed retrieval Query messages. Here peers receive QueryHit messages requested by them, Query messages and QueryHit messages to the Query messages requested by other peers. Yet in the internet(WWW) where a great deal of service requests and retrieval processes are carried out, the relation between retrieval hit frequency, which is connection frequency, and ranking value for the frequency follows the zipf's Law. Therefore web requests follow the zipf's distribution and zipf's law can be applied with retrieval request Query and QueryHit messages collected from Gnutella network. With standard keyword of collected Query message and retrieved share file name of Query message, approximate constant can be obtained through zipf's law. Namely regularized approximate constant can be obtained with standard keyword from Query message and share file name retrieved from QueryHit message by ranking the frequency and periodically updating this. As shown on Fig 1, 2, the amount of with TTL value, 1 Query message is the high but more than 70% of the messages are discarded so with TTL value, 2 Query message is valid. Accordingly most of QueryHit message, retrieval respond value, receives messages with TTL value, 3 and peers that relayed to messages with TTL value, 2 is transferred. Therefore to apply zipf's law in the current participating Gnutella network, it is the most efficient to apply TTL value, 2. And peers that receive Query messages with TTL value, 2 transfer QueryHit message, most of the retrieval respond value so file in peer receiving Query with TTL value, 2 has the highest possibility of the final service file desired by user who requested retrieval. Fig. 3 show that the result of broadcasting one packet once for one host and based on Newtella Gnutella Clone on Gnutella protocol basis. Fig. 3 simulates message distribution based on of average peers(about 100,000 peers) of top 20 servers at <http://ed2k.2x4u.de/list.html> managing server and user list at eDonkey,

which is a P2P file share program. But if only 1/10 of those peers are sharing files and using Gnutella protocol, we can search about 10,000 useful hosts

In Fig. 5 when TTL value is 1 more than 99.7% is discarded messages and when 2, the percentage is similar to those with value 1 but the valid messages are those with TTL value 2 and 3. However since TTL value 2 makes great percentage per total package rather than TTL value 3, Query message with TTL value 2 discovers what the most popular keyword is in the current network and user may choose retrieval result of QueryHit message transferred by peer that receives Query with TTL value 2 that it is more efficient to cache files having TTL value 2 for message result value.

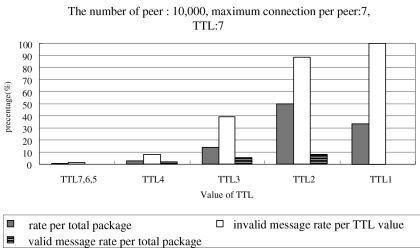


Fig. 3. Message ratio according to TTL value

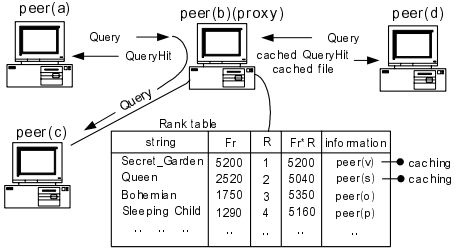


Fig. 4. Caching technique based on zipf's law

In Fig. 4 peer(b) forms table with retrieval standard keyword of Query message with TTL 2 according to zipf's Law. If  $\text{ranking} \times \text{frequency} (R \times Fr)$  value reaches to approximate constant value peer(b) periodically caches and updates relaying Query-Hit messages related to high ranking since high ranking values are virtually the most popular keywords in Gnutella network and in some cases it can cache file. When peer (d) requests relevant high ranking retrieval to peer(b) cached QueryHit messages are transferred as respond value and if there is cached file the relevant share resource information is transferred as QueryHit message. Also since contents service quality of peer to provide information can be known through QueryHit message it can be more effective to cache it in case that the service quality is lower than its resource.

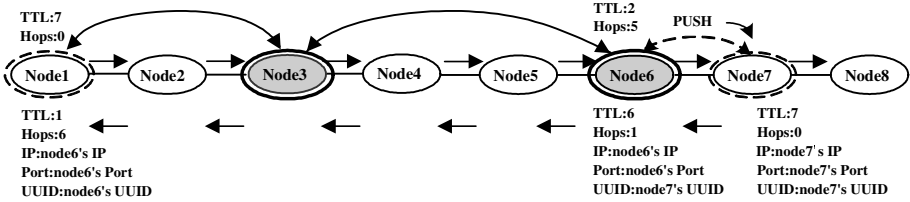
## 4 Providing Server and Client Anonymity

To hide the client and the server from each other, our algorithm selects a limited number of random agent nodes and let them relay the actual data between the server the client. The agents are selected dynamically per each session, hence it is hard to trace the location of the server or the client via the agent nodes. Also our algorithm makes sure the number of selected agents is very limited, in most cases one or two, so that the relay overhead can be minimized.

When the incoming packet is a QueryHit packet, each node performs an algorithm to determine whether it is going to be an agent for this query. It generates a random number in  $[0..\text{PATH\_LEN}-1]$ . PATH\_LEN is the length of the communication path

this query belongs to and can be computed by  $TTL + Hops - 1$  in the received packet. When added, TTL and Hops in the QueryHit packet represents the length of the path between the client and the server; by choosing a random number in  $[0, PATH\_LEN-1]$ , we hope approximately one node in the communication path will establish itself as an agent node. If the random number is 0, it becomes an agent. There is a chance no node in the path will pick 0; in this case there will be no agent node. There is also a chance that multiple nodes pick 0's; then we have multiple agent nodes. Multiple agent nodes will increase the relay-overhead but will strengthen the degree of anonymity as well. Failure to select an agent node will expose the client and the server to each other; but both still don't know for sure that the other part is the actual server or client.

The agent node will replace the IP Address and Port in the QueryHit packet with its own ones. The ServantSessionUUID remains the same. Now this packet will have the IP Address and Port of the agent instead of the original server, but it still has the ServerSessionUUID of the original server. The client, after receiving this packet, will contact the specified IP Address with a wrong UUID, and this discrepancy will be detected by our agent node allowing it to act as a proxy between the client and the server. For this purpose, the information about the original server is saved in the SessionTable of the agent node as shown in the algorithm. The same algorithm is repeated at following agent nodes when there are more than one agent. For the second agent, the first agent will be treated as the original server, and it will establish itself as a proxy between the client and the first agent node. This argument goes on for the third and fourth agent nodes as they are added in the path. Note we are not adding all intermediate nodes between the client and the server as agent nodes as in MUTE; we are selecting only those who picks 0.



**Fig. 5.** Selection of agent nodes and Flow of Query and Query Hit packets

Fig. 5 shows a typical flow of a Query and QueryHit packet. Node 1 broadcasts a Query packet for which Node 7 has the requested data. Node 7 sends a QueryHit packet back to Node 1 via the same path that the Query packet followed. Without anonymity mechanism, the client can contact Node 7 directly since the QueryHit packet contains the IP Address and Port of Node 7. Node 7, on the other hand, also contacts node 1 directly to answer the request; hence both are exposed to one another.

In our scheme, some agent nodes will be elected as the QueryHit packet traces back to Node 1. Suppose Node 6 and Node 3 are such agent nodes. Upon deciding to become an agent, Node 6 starts to modify the packet header: the IP Address and Port

of Node 7 are replaced with those of Node 6. And the related information is saved in the SessionTable. Node 6 now acts as if it is the server who has sent the QueryHit packet. Node 3 also processes the packet similarly, but in this case, it thinks Node 6 is the server and sends the modified packet to Node 1 as a proxy for Node 6. Node 1, upon receiving QueryHit packet, contacts Node 3, the first agent, and sends HTTP header to it requesting data. The UUID included in the packet, however, is that of Node 6. Node 3 knows that this packet should be delivered to Node 6 by noticing this mis-matching between UUID and the destination IP address. It builds a PUSH packet to request the indicated data from Node 6. Now Node 3 works as an agent between Node 1 and Node 6. The response from Node 6 will be relayed to Node 1 via Node 3. Similar events happen in Node 6 because it is another agent between Node 3 and the final server. Two agents, Node 3 and Node 6, will relay the data from the actual server to the client hiding the identities of both ends successfully

## 5 Experimentation and Evaluation

To prove the effectiveness of our algorithm, we have built a simulated P2P network with Minism[27]. Minism allows us to generate a random topology for a number of nodes and to simulate packet broadcasting. Things such as what percentage of the original packets reaches the final destination or how many packets are discarded during the propagation can be answered. We have modified the behavior of Minism to implement our algorithm. Especially a routing table is built to trace the movement of QueryHit packet. Fig. 6 shows the inner-working of Minism code. Fig. 6 shows the propagation of Query packet. The "reached" array shows the nodes the Query packet reached at each stage. In "reached" array, nodes with -1 are ones that are not reached yet; nodes with 1 are those that are visited already; finally a node with (1) is one we are going to visit next. Below "reached" array, we can see the state of the stack that contains the Query packet. To simulate the propagation of a packet, the stack shows at each stage which path each duplicated Query packet should follow (from which node to which node). For example, at stage 1, all nodes are marked with -1 except node 0 since we haven't visited any node yet, and the starting node is node 0. The stack contains the path we should relay the packet: from -1 (no where) to 0 (the starting node). At stage 1, we can see node 0 is already visited (at stage 1); the next node we should visit is node 3 because node 0 has two neighbors - node 3 and 4 - and node 3 is selected as the next one. The stack shows the two path segments we should take (from 0 to 3 and from 0 to 4) for the Query packet. The segment at the top of the stack (from 0 to 3) will be chosen.

To implement our algorithm, we have included routing tables that show at each stage which path the Query packet has followed. Fig. 7 shows such tables. At stage 1, the first table shows route(0,3) and route(0,4) has been taken. At stage 2, it shows that route(3,5), route(3,9999), route(4,7), etc. has been taken. With these tables, we can backtrack the paths starting from the destination all the way back to the original client.

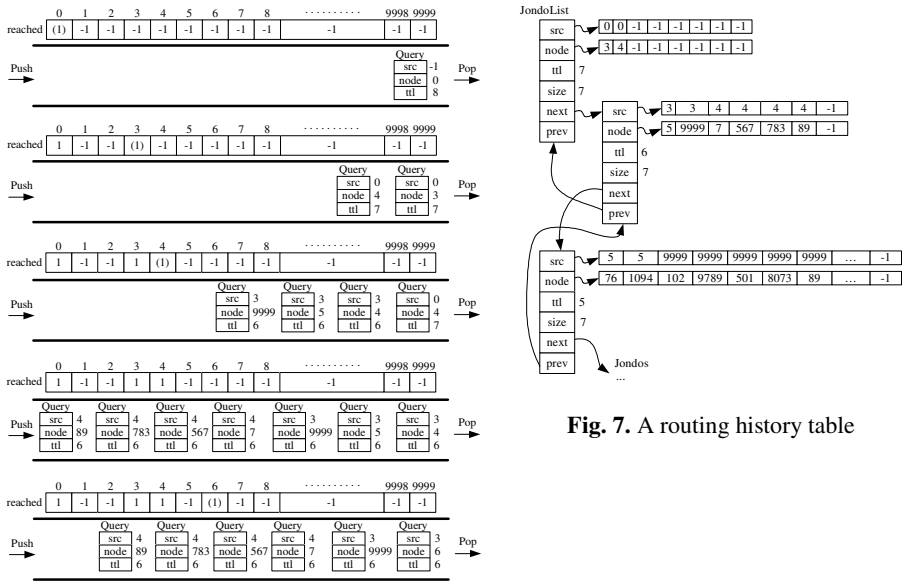
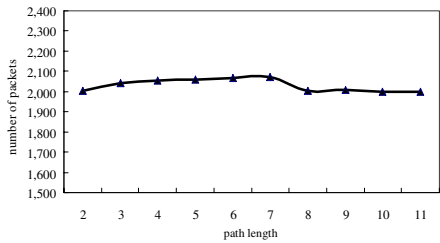


Fig. 6. The propagation of a Query packet

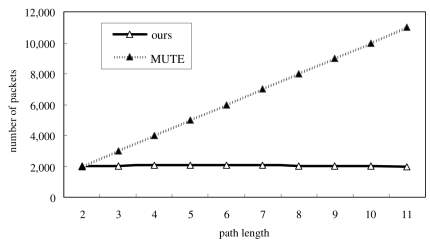
Fig. 7. A routing history table

The result is shown in Fig. 8 and 9. Fig. 8 shows the number of packets generated for our simulation. Path length between the client and the server is chosen to vary from 2 to 11. Fig. 9 shows the performance of our algorithm compared to that of MUTE. As expected MUTE increases the traffic linearly as the path length gets longer, while ours stays almost the same. In most cases only one node chooses to be an agent. Since we do not allow any additional controlling packet to relay the information about this selection process, there is a chance that no node will decide to become the agent or all nodes become the agents. The latter case will make our algorithm to mimic that of MUTE, the former to that of no anonymity case. It could be a problem if there are no node selected as an agent: the client and the server is not hidden from each other. But as explained in the previous section, with our scheme the client or the server never knows that it is dealing with the other side directly. The chance is very small and each side cannot attack the other with such a small percent of certainty. As shown on Fig. 1 and Fig. 2 Message of TTL value 1 amounts approx. 34% and 33%, which is 99.92% of that, is to be discarded as invalid message in simulation with 10,000 peers, 7 connectable peers and TTL value 11. Fig. 10 show that the result of broadcasting one packet once for one host and based on Newtella Gnutella Clone[11,12] on Gnutella protocol basis. Fig. 7 shows the result of applying zipf's law with 7 of maximum connection per each peer, and TTL value 10. Result from previous section indicates that with TTL value 1 many Query messages amounting to about 49.83% were reduced and about 17% was reduced in total. Fig. 11 show that the result of broadcasting one packet once for one host. Fig. 11 simulated message throughput rate by number of peer and Query message frequency having TTL value 1. The result shows that message throughput rate which indicates all the Query messages sent by non-specific peer were transferred to all the currently participating peers was

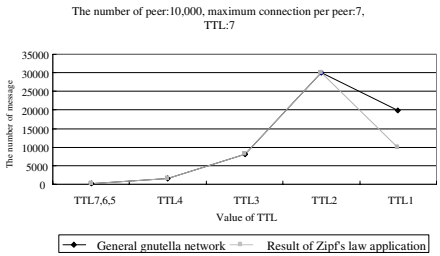
constantly remained as 100%(in 10,000 peers). Although large amount of broadcasted messages were reduced, messages were transferred to all peers. It means that request sent by peer is able to retrieve every matching data among all peers participating network. Therefore retrieval result which can be obtained by user is constant. And the message throughput rate was not changed to the result applied caching technique proposed at this study whereas message amount was considerably reduced instead. This shows that despite the reduction of message amount it hardly affect the retrieval result.



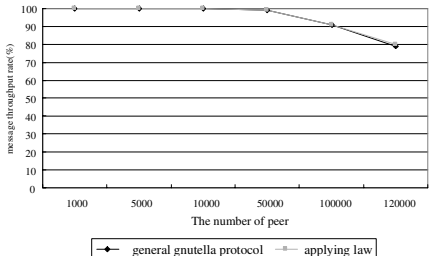
**Fig. 8.** Number of packets generated for each path length



**Fig. 9.** Comparison between MUTE and our technique in terms of generated traffic to deliver the contents



**Fig. 10.** Simulation results of application



**Fig. 11.** Simulation results of Message throughput rate

## 6 Conclusions

In this paper, we have proposed a protocol for anonymous file sharing protocols and caching technique in P2P file sharing systems. Currently, many techniques are suggested to provide anonymity for the client or the server. We proposed a technique that stood about in the middle of these two approaches. We employed the idea of a proxy but not static one. The proxy is selected dynamically during the traverse of the QueryHit packet. Since the selection process is truly distributed, no one knows exactly how many proxies, or agents, are selected and where they are located. The agents are linked together only by neighbors: each agent knows only its previous and the succeeding one. We have designed the process such that only very limited number of agents are selected. Since these agents are used only for the current session, it is hard to attack them or to predict the location of the client or the server by analyzing their

packets. And this study proposes keyword and retrieval file caching technique Which effectively reduces lots of broadcasted Query messages transferred between agents. Caching policy will be established using correlation with keyword of user's Query message, frequency and ranking of responding strings retrieved and the size of retrieved files. By Caching them according to the policy and spreading the caching policy to neighboring agents, the overall P2P network performance can be improved. We explained how it works and showed that its performance improved substantially over previous techniques through experimentation.

## Acknowledgements

This work was supported by INHA UNIVERSITY Research Grant.

## References

1. I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, Freenet: A distributed anonymous information storage and retrieval system, In Workshop on Design Issues in Anonymity and Unobservability, pages 46.66, 2000., <http://citeseer.nj.nec.com/clarke00freenet.html>.
2. The Gnutella Protocol Specification v0.41 Document Revision 1.2., <http://rfcgnutella.sourceforge.net/developer/stable/index.html/>
3. Kirsten Hildrum, John Kubiawicz, Satish Rao and Ben Y. Zhao, Distributed Object Location in a Dynamic Network, Theory of Computing Systems (2004)
4. Roger Dingledine, Nick Mathewson, Paul Syverson, Tor: The Second Generation Onion Router, Proceedings of the 13th USENIX Security Symposium (2004)
5. Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph, and John Kubiawicz, Tapestry: A Resilient Global-scale Overlay for Service Deployment, IEEE Journal on Selected Areas in Communications (2004)
6. Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, Hari Balakrishnan, Chord: a scalable peer-to-peer lookup protocol for internet applications, IEEE/ACM Transactions on Networking (2003)
7. Neil Daswani, Hector Garcia-Molina, Query-flood DoS attacks in gnutella, Proceedings of the 9th ACM conference on Computer and communications security table of contents (2002)
8. P. Krishna Gummadi, Stefan Saroiu, Steven D. Gribble, A measurement study of Napster and Gnutella as examples of peer-to-peer file sharing systems, ACM SIGCOMM Computer Communication Review (2002)
9. A. Back, U. Möller, and A. Stiglic, Traffic analysis attacks and trade-offs in anonymity providing systems, In I. S. Moskowitz, editor, Information Hiding (IH 2001), pages 245.257. Springer-Verlag, LNCS 2137, 2001.
10. J. F. Raymond, Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems, In Workshop on Design Issues in Anonymity and Unobservability. Springer-Verlag, LNCS 2009, July 2000.
11. Anonymizer, <http://www.anonymizer.com/>
12. E. Gabber, P. Gibbons, D. Kristol, Y. Matias, and A. Mayer, "Consistent, yet anonymous, Web access with LPWA," Commun. ACM, vol.42, no.2, pp.42?47, Feb. 1999.
13. M.K. Reiter and A.D. Rubin, "Crowds: Anonymity for Web transactions," ACM Trans. Information and System Security, vol.1, no.1, pp.66?92, Nov. 1998.



14. D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Commun. ACM*, vol.24, no.2, pp.84-90, 1981.
15. C. Shields and B.N. Levine, A protocol for anonymous communication over the Internet, *Proc. 7th ACM Conference on Computer and Communication Security (ACM CCS 2000)*, Nov. 2000.
16. D.L. Chaum, Untraceable electronic mail, return addresses, and digital pseudonyms, *Communications of the ACM* 24(2), 84-88 (1981).
17. L. Cottrell, Frequently asked questions about Mixmaster remailers, <http://www.obscura.com/~loki/remailer/mixmaster-faq.html> (2000).
18. Zero-Knowledge Systems, <http://www.zks.net/> (2000).
19. B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, Private information retrieval, *Journal of the ACM* 45(6), 965-982 (1998).
20. The Rewebber, <http://www.rewebber.de/> (2000).
21. M. Waldman, A.D. Rubin, and L.F. Cranor, Publius: a robust, tamper-evident, censorship-resistant, web publishing system, in *Proceedings of the Ninth USENIX Security Symposium*, Denver, CO, USA (2000).
22. R.J. Anderson, The Eternity service, in *Proceedings of the 1st International Conference on the Theory and Applications of Cryptology (PRAGOCRYPT '96)*, Prague, Czech Republic (1996).
23. R. Dingledine, M.J. Freedman, and D. Molnar, The Free Haven project: distributed anonymous storage service, in *Proceedings of the Workshop on Design Issues in Anonymity and Unobservability*, Berkeley, CA, USA. Springer: New York (2001).
24. MUTE: Simple, Anonymous File Sharing., <http://mute-net.sourceforge.net/>
25. Michael J. Freedman, Robert Morris, Tarzan: A Peer-to-Peer Anonymizing Network Layer, in *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, Cambridge, MA, USA (2002)
26. Stephen C. Bono, Christopher A. Soghoian, Fabian Monrose, Mantis: A Lightweight, Server-Anonymity Preserving, Searchable P2P, Information Security Institute of The Johns Hopkins University, Technical Report TR-2004-01-B-ISI-JHU (2004)
27. Gnutella Developer Forum., [http://groups.yahoo.com/group/the\\_gdf/](http://groups.yahoo.com/group/the_gdf/)
28. C. Shirky, Listening to Napster, in *Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology*, ed. A. Oram, O'Reilly and Associates, Inc., Sebastopol, California, 2001.
29. V. Scarlata, B.N. Levine, and C. Shields, Responder anonymity and anonymous peer-to-peer file sharing, *Proc. IEEE Intl. Conference on Network Protocols (ICNP) 2001*, Nov. 2001.
30. Introduction to Newtella, <http://www.schnarff.com/gnutelladev/source/newtella/>
31. Lee Breslau, Pei Cao, Li Fan, Graham Phillips, Scott Shenker, Web Caching and Zipf-like Distributions: Evidence and Implications, *IEEE INFOCOM*, MONTH.1999.
32. Kunwadee Sripanidkulchai, The popularity of Gnutella queries and its implications on scalability, <http://www-2.cs.cmu.edu/~kunwadee/research/p2p/paper.html>, JAN. 2004.

# Conjunctive Queries over DAGs

Royi Ronen and Oded Shmueli\*

Technion, Israel Institute of Technology  
{ronenr, oshmu}@cs.technion.ac.il

**Abstract.** We address the complexity of evaluating conjunctive queries over labeled directed acyclic graphs (DAGs). DAG-structured data and queries over such data occur in many contexts, such as ontologies for the Semantic Web and Complex Event Processing. The relations representing the DAG are binary axis relations, mostly adopted from XPath, and unary label relations. The relations definitions are generalized for the DAG case. We prove that all polynomial time results of [6], except one, become NP-Complete over DAGs.

## 1 Introduction

The complexity of conjunctive queries over tree structure data, a problem motivated primarily by XML-related technologies was characterized by Gottlob and Schulz in [6]. In this paper we study the complexity of conjunctive queries over data which is DAG-structured rather than tree-structured, a problem which occurs in many areas of modern information technology. We elaborate on two of these areas in section 1.1.

In [6], the relations representing the tree-structured originate from XPath axes, and conjunctive queries use these relations as predicates. We start our study by generalizing the problem of conjunctive queries over trees to the DAG case by generalizing the semantics of relations so that they represent DAG-structured data rather than tree-structured data. Then, we present the methods used in [6] to prove both hardness and tractability results for the tree case. We build on these methods in our proofs as follows. We use the notion of hemichordality with respect to a total order in order to prove that one fragment which is tractable in the tree case remains tractable in the DAG case. We prove hardness results by extending the reduction used to prove hardness results on trees. We show that while many fragments (including every fragment that consists of no more than one relation) are polynomial in the tree case, every possible fragment except one becomes NP-Complete in the DAG case, even if consists of just one relation.

### 1.1 Motivation

**Semantic Web.** The Semantic Web is a vision for the future of the World Wide Web according to which the organization of data is according to meaningful concepts. Semantics is expected to dramatically improve the efficiency of knowledge

---

\* O. Shmueli was funded in part by ISF grant 890015 and by the P. and E. Nathan Research fund.

management because data will become more meaningful to machines [10], that will be able to assert the correctness of existing data and extract new knowledge out of existing knowledge [4].

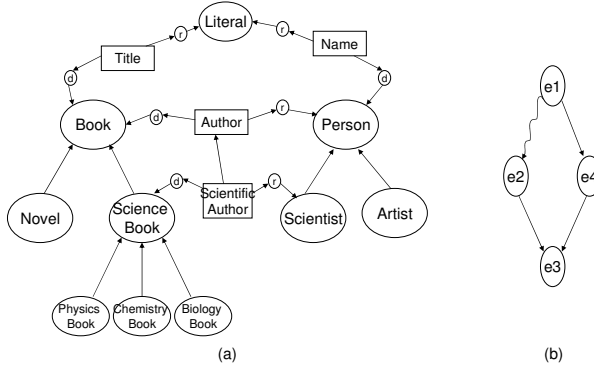
Ontologies are a key concept in the Semantic Web which provide a means to classify and name objects of the modeled domain, their properties and the relationships among them. A crucial step in defining an ontology is the organization of the relevant terms in a taxonomic hierarchy [4], i.e., stating a *subclass* relationship between classes of objects, which means that if  $X$  is a subclass of  $Y$ , then every object in  $X$  is also in  $Y$ . A *DAG taxonomy* is a taxonomy where a class may be a subclass of one class or more, as opposed to a *tree taxonomy* where a class can be a subclass of at most one class. The Resource Description Framework Schema (RDFS) [9] is a framework for constructing ontologies. Classes in RDFS ontologies are organized in a DAG taxonomy (cycles imply that participating classes are equivalent and can be eliminated from the graph). A Class may be assigned properties explicitly or inherit properties from a superclass. Properties themselves may also be organized as a DAG taxonomy. The information in the ontology provides semantics to data in Resource Description Framework (RDF) [9] instances which refer that ontology. This is done by RDF elements *subjects*, *objects* and *predicates* referring to RDFS elements as types.

We use an example to demonstrates the use of conjunctive queries over DAGs in the Semantic Web context. Figure 1(a) depicts an ontology for an electronic bookstore. The convention we use to represent the ontology as a graph is taken from [4] with a slight technical modification. An ellipse represents a class and a rectangle represents a property. A directed edge  $(u, v)$  where  $u$  and  $v$  are both classes (respectively, properties) means that  $u$  is a subclass (respectively, subproperty) of  $v$ . While in [4] edges labeled by  $d$  (for domain) and  $r$  (for range) connect a property to its domain and range classes of values, we simulate this edge label by splitting the edge into two edges and adding a new node, labeled either  $d$  or  $r$ . One edge connects the property to the new node and the other edge connects the new node to the class. Therefore, edges in our graph are label-free. The ontology in Figure 1(a) specifies the following taxonomy: Chemistry Book, Physics Book and Biology Book are subclasses of Scientific Book. The latter and Novel are each a subclass of Book. Person is a superclass of Scientist and of Artist. The figure does not show that all classes are subclasses of Class, which is the class of all classes in RDFS.

For example, Chemistry Book inherits the properties "Scientific Author", "Author" and "Title" from its superclasses. The following conjunctive query (whose conjuncts' semantics have not yet been formally defined) extracts from the graph all the classes in the ontology that are in the range of the property "Author", explicitly or by inheritance.

$$Res(X) \leftarrow Child^*(X, W), Child(P, R), Label_{Author}(P), r(R), Child(R, W), Class(X).$$

Note that arrows between classes are in the leaf to root direction. After the evaluation of this conjunctive query, the relation  $Res(\cdot)$  will contain all the classes in the ontology that may be in the range of the property "Author" in an RDF



**Fig. 1.** (a) Electronic bookstore ontology. (b) Event pattern expressed in the conjunctive query. A straight arrow represents direct causality, and a curly arrow represents causality (possibly) through other events.

instance which refers to the above RDFS (*Class*( $\cdot$ ) denotes a relation whose elements are all the classes).

**Causal Event Histories.** The area of Complex Event Processing (CEP) deals with tracking the causality between events, i.e., which event(s) caused additional complex events (also known as "situations" [2]) to occur. CEP has many applications ranging from low level infrastructure management to high level business intelligence. The log of the history of events can be represented as a DAG in which every node represents an instance of one event type, and an edge  $(u, v)$  exists if  $u$  caused  $v$  (solely or together with other events) [7]. This DAG does not represent rules that define triggering relationships between event types, but represents what actually happened in the CEP system in terms of instances of events. Conjunctive queries over DAGs can be a powerful root-cause analysis tool and a means to gather information about the modeled reality. For example, the following Boolean query checks if the pattern in Figure 1(b) exists in a causal event history.

$Q() \leftarrow e1(X), Child(X, Y), e4(Y), Child^*(X, Z), e2(Z), Child(Y, W), Child(Z, W), e3(W).$

The pattern means that  $e1$  caused  $e4$  directly.  $e1$  also caused a possibly empty sequence of events followed by an  $e2$  event.  $e2$  and  $e4$  together caused  $e3$ .

## 1.2 Related Work

Gottlob and Schulz [6] characterize the complexity of conjunctive queries over trees. The relations which represent the data are axis-relations adopted from XPath axes and label relations over a labeling alphabet  $\Sigma$ . A *signature* is a

set of relations, and a conjunctive query over a signature  $s$  is a conjunctive query whose conjuncts are taken only from  $s$ . The following table summarizes the complexity results:

| <i>Signature</i>  | <i>Complexity Class</i> |
|---|-------------------------|
| $\{Child^*(\cdot, \cdot), Child^+(\cdot, \cdot), Label_a(\cdot)_{a \in \Sigma}\}$   | P                       |
| $\{Following(\cdot, \cdot), Label_a(\cdot)_{a \in \Sigma}\}$  | P                       |
| $\{Child(\cdot, \cdot), NextSibling(\cdot, \cdot), NextSibling^+(\cdot, \cdot), NextSibling^*(\cdot, \cdot), Label_a(\cdot)_{a \in \Sigma}\}$ | P                       |
| Every signature not included in one of the above signatures   | NP-Complete             |

Polynomial results are w.r.t. combined complexity [1, 8] and NP-Completeness results are w.r.t. query complexity (i.e., expression complexity) [1, 8].

### 1.3 Contributions

We generalize the definition of axis relations, originating in XPath queries over trees, to the DAG case. We characterize the complexity of conjunctive queries over DAGs. Our results appear in the following table.

| <i>Signature</i>   | <i>Complexity - DAG</i> | <i>Complexity - Tree</i> |
|--|-------------------------|--------------------------|
| $\{Child^+(\cdot, \cdot), Label_a(\cdot)_{a \in \Sigma}\}$       | NP-Complete             | P                        |
| $\{Child^*(\cdot, \cdot), Label_a(\cdot)_{a \in \Sigma}\}$       | NP-Complete             | P                        |
| $\{Child(\cdot, \cdot), Label_a(\cdot)_{a \in \Sigma}\}$         | NP-Complete             | P                        |
| $\{Following(\cdot, \cdot), Label_a(\cdot)_{a \in \Sigma}\}$     | P                       | P                        |
| $\{NextSibling(\cdot, \cdot), Label_a(\cdot)_{a \in \Sigma}\}$   | NP-Complete             | P                        |
| $\{NextSibling^+(\cdot, \cdot), Label_a(\cdot)_{a \in \Sigma}\}$ | NP-Complete             | P                        |
| $\{NextSibling^*(\cdot, \cdot), Label_a(\cdot)_{a \in \Sigma}\}$ | NP-Complete             | P                        |

Here too, polynomial results are w.r.t. combined complexity and NP-Completeness results are w.r.t. query complexity.

## 2 Preliminaries

We represent the data using relations as follows. The relations representing the structure of the DAG, originating from XPath, are adopted from [6]:  $\{Child(\cdot, \cdot), Child^+(\cdot, \cdot), Child^*(\cdot, \cdot), Following(\cdot, \cdot), NextSibling^+(\cdot, \cdot), NextSibling(\cdot, \cdot), NextSibling^*(\cdot, \cdot)\}$ . When discussing trees, the semantics of the first five relations is identical to the semantics of the corresponding XPath [11] axes  $\{child, descendant, descendant-or-self, following, following-sibling\}$  respectively.  $NextSibling(\cdot, \cdot)$  corresponds to  $following-sibling[position() = 1]$  and  $(x, y) \in NextSibling^*(\cdot, \cdot)$  iff  $(x, y) \in NextSibling^+(\cdot, \cdot) \cup \{(x, x)\}$ . Note that this is a superset of XPath, since every axis is either covered above or covered implicitly by its reverse axis. For example, a relation  $Parent(\cdot, \cdot)$  with semantics corresponding to the XPath axis "parent" can be simulated by the  $Child(\cdot, \cdot)$  relations with swapped arguments.

**Generalization to DAGs.** In the case of DAGs, these relations remain a natural way to express conjunctive queries, but a generalization of the definition for some of them is required. Following are definitions of the axis relations, which represent the DAG structure. We denote by  $(V, E)$  the directed graph corresponding to the DAG expressed by the relations. Since we discuss ordered graphs, The schema of  $E$  is  $E(X, Y, N)$  and elements are of the form  $(x, y, n)$  where  $x$  is the source of the edge,  $y$  is the destination and  $n$  is the number of  $y$ 's position among  $x$ 's outgoing edges from left to right.

- $(x, y) \in \text{Child}(\cdot, \cdot) \Leftrightarrow \exists n$  such that (s.t.)  $(x, y, n) \in E$ .
- $(x, y) \in \text{Child}^*(\cdot, \cdot) \Leftrightarrow (x, y) \in (\pi_{x,y} E)^*$ , where  $R^*$  denotes the reflexive and transitive closure of the relation  $R$  (i.e., reachable via edges in zero or more steps).
- $(x, y) \in \text{Child}^+(\cdot, \cdot) \Leftrightarrow (x, y) \in (\pi_{x,y} E)^+$ , where  $R^+$  denotes the transitive closure of the relation  $R$  (i.e., reachable via edges in one or more steps).
- $(x, y) \in \text{Following}(\cdot, \cdot) \Leftrightarrow x$  precedes  $y$  in the postorder [3] numbering where the children of a node are explored according to their position relative to their siblings, and  $(y, x) \notin \text{Child}^*(\cdot, \cdot)$ . If there is more than one root, roots are explored in the order of their object IDs, a unique value that exists for every object.
- $(x, y) \in \text{NextSibling}^+(\cdot, \cdot) \Leftrightarrow \exists v \in V \text{ s.t. } ((v, x, n) \in E \wedge (v, y, m) \in E \wedge (m > n))$ .
- $(x, y) \in \text{NextSibling}^*(\cdot, \cdot) \Leftrightarrow \exists v \in V \text{ s.t. } ((v, x, n) \in E \wedge (v, y, m) \in E \wedge (m \geq n))$ .
- $(x, y) \in \text{NextSibling}(\cdot, \cdot) \Leftrightarrow \exists v \in V \text{ s.t. } ((v, x, n) \in E \wedge (v, y, n+1) \in E)$ .

Nodes are labeled. We denote the labeling alphabet by  $\Sigma$ , and allow multiple labels for the sake of simplicity of proofs. The tractability result remains valid for singly labeled graphs. After proving NP-Hardness, we show how multiple labels can be eliminated from each of the proofs.

## 2.1 Polynomial Results for Conjunctive Queries over Trees

We briefly summarize the methodology used in [6] to prove polynomial time results for trees as it is used for the DAG case. Let  $R$  be a binary relation in a relational structure  $S$  and  $<$  be a total order on  $S$ 's domain,  $A$ . Then, we call  $R$  *<-hemichordal* if for all  $n_0, n_1, n_2, n_3 \in A$  s.t.  $n_0 < n_1$  and  $n_0 \leq n_2 \leq n_3$ ,  $R(n_1, n_2) \wedge R(n_0, n_3) \rightarrow R(n_0, n_2)$  and  $R(n_2, n_1) \wedge R(n_3, n_0) \rightarrow R(n_2, n_0)$ .

**Lemma 1.** *Let  $A$  be the finite domain of values for a relational structure  $S$ ,  $R$  be a binary relation on  $S$  and  $<$  be a total order on  $A$  s.t.  $R \subseteq \leq$ . Then,  $R$  is <-hemichordal iff for all  $n_0, n_1, n_2, n_3 \in A$  s.t.  $n_0 < n_1 \leq n_2 < n_3$ ,  $R(n_1, n_2) \wedge R(n_0, n_3) \rightarrow R(n_0, n_2)$ .*

Lemma 1 provides a means to prove hemichordality in the case where the relation is a subset of the total order with respect to which one tries to prove hemichordality.

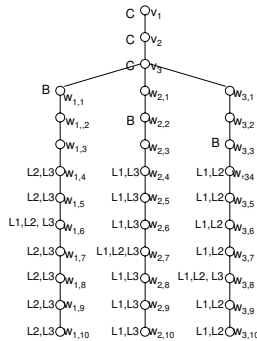
**Lemma 2.** *Let  $S$  be a relational structure of unary and binary relations. The evaluation of a Boolean conjunctive query  $Q$  over a structure  $S$  in which all binary relations are  $<$ -hemichordal has polynomial time complexity w.r.t. combined complexity.*

The evaluation of a  $k$ -ary conjunctive query (a query with  $k$  variables in the query's head) can be done by evaluating  $|A|^k$  Boolean conjunctive queries, where  $|A|$  is the size of the finite domain of values for the structure under discussion. Therefore, both tractability and NP-Hardness results which hold for Boolean conjunctive queries, hold for  $k$ -ary conjunctive queries as well. Proofs for the claims in section 2.1 can be found in [6], which uses them in order to prove the polynomial results in 1.2 by showing that each of the three signatures in 1.2 (P) contains unary relations and binary relations which are  $<$ -hemichordal w.r.t. the same total order  $<$ .

## 2.2 NP-Completeness Results for Conjunctive Queries over Trees

We briefly show the reduction used to prove the NP-Completeness of conjunctive queries over trees over the signature  $\{Child(\cdot, \cdot), Child^+(\cdot, \cdot), Label_a(\cdot)_{a \in \Sigma}\}$ , as we use this later on, by reducing it in order to prove NP-Hardness of the problems under discussion in this paper. *1-in-3 SAT*<sup>+</sup> [5] is the NP-Complete problem of determining if there exists a truth assignment for the variables of a 3SAT instance  $I$  with only positive literals such that every clause in  $I$  has exactly one true literal out of its three different literals. Denote  $I = C_1, \dots, C_m$  where  $C_i$  represents the disjunction of three positive literals. The reduction polynomially builds a conjunctive query based on this instance, which evaluates to **true** on the fixed tree in Figure 2 iff there is a satisfying truth assignment for  $I$ .

In the tree in Figure 2, every node appears with an ID and some of the nodes appear with one or more labels. Labels are capitalized, IDs are not. For example, the node with ID  $w_{3,9}$  is labeled by  $L1$  and  $L2$ . Edges represent the  $Child(\cdot, \cdot)$  relation between nodes. The query is built as follows. For  $1 \leq i \leq m$ , the new



**Fig. 2.** Tree used for proving NP-Hardness of  $\{Child(\cdot, \cdot), Child^+(\cdot, \cdot), Label_a(\cdot)_{a \in \Sigma}\}$  over trees

variables  $x_i$  and  $y_i$  are introduced. Whenever the  $k$ -th literal of  $C_i$  coincides with the  $l$ -th literal of  $C_j$  ( $1 \leq i, j \leq m$ ,  $1 \leq k, l \leq 3$ ,  $i \neq j$ ), a new variable  $z_{k,l,i,j}$  is introduced. The query's conjuncts are:

- for  $1 \leq i \leq m$ , the conjuncts  $C(x_i)$ ,  $B(y_i)$ ,  $Child^3(x_i, y_i)$  are included, where for  $k > 1$ ,  $A^k(s_0, e_0)$  stands for  $A(s_0, n_0), A(n_0, n_1), \dots, A(n_{k-2}, e_0)$  where the  $n_i$  are new and distinct variables.
- for each variable  $z_{k,l,i,j}$ , the three conjuncts  $L_k(z_{k,l,i,j})$ ,  $Child^+(y_i, z_{k,l,i,j})$  and  $Child^{8+k-l}(x_i, z_{k,l,i,j})$  are included.

The basic idea used in our proofs is to modify the tree into a DAG (by deleting original edges and adding new nodes and edges) and provide a new Boolean query that evaluates to **true** on the DAG iff the above query evaluates to **true** on the tree.

### 3 Complexity Results for Queries over DAGs

#### 3.1 Polynomial Result for Conjunctive Queries over DAGs

Consider the postorder numbering of the nodes in the DAG. If more than one root exists, roots are explored in the order of their object IDs. Let  $<$  be the total order corresponding to this postorder numbering. It is easy to see that  $\{(x, y) \text{ s.t. } Following(x, y)\} \subseteq \{(x, y) \text{ s.t. } x < y\}$ .

**Lemma 3.** *Following( $\cdot, \cdot$ ) is  $<$ -hemichordal.*

*Proof.*  $Following(\cdot, \cdot) \subseteq <$ , therefore we can use lemma 1. Assume that  $n_0, n_1, n_2, n_3$  are nodes in the DAG such that  $n_0 < n_1 \leq n_2 < n_3$ , and that  $Following(n_1, n_2)$  and  $Following(n_0, n_3)$ .  $n_0 < n_1$  entails exactly one of the following:

- $Child^+(n_1, n_0)$ , which together with  $Following(n_1, n_2)$  entails  $Following(n_0, n_2)$  by definition.
- $Following(n_0, n_1)$ , which together with  $Following(n_1, n_2)$  entails  $Following(n_0, n_2)$  since  $Following(\cdot, \cdot)$  is transitive.

No other options exist since  $<$  equals to the union of the inverse of  $Child^+(\cdot, \cdot)$  and  $Following(\cdot, \cdot)$ . This proof is very similar to the proof of  $<$ -hemichordality of  $Following(\cdot, \cdot)$  in [6], however our definition of the relation  $Following(\cdot, \cdot)$  is different in the DAG case.  $\square$

**Theorem 1.** *Conjunctive queries over the signature  $\{Following(\cdot, \cdot), Label_a(\cdot)_{a \in \Sigma}\}$  are in  $P$  w.r.t. combined complexity.*

*Proof.* Using lemma 2, we conclude that the problem is in  $P$ .  $\square$



### 3.2 NP-Completeness Results for Conjunctive Queries over DAGs

#### NP

**Theorem 2.** *The problem of evaluating a Boolean conjunctive query over a DAG, which may use all the axis or label relations, is in NP w.r.t. query complexity (i.e., with a fixed database).*

*Proof.* Consider a conjunctive query  $q$  having  $Var(q)$  distinct variables in it. A non-deterministic Turing machine can guess a node for each variable, and check for each conjunct whether or not it evaluates to **true** under that assignment. Guessing the nodes requires polynomial time. We now show that checking the truth value for each conjunct can be done in polynomial time.

- $Child(x_0, y_0)$ . Checking if  $x_0$  is one of the parents of  $y_0$  is  $O(|E|)$  operations, simply by checking all of  $x_0$ 's children.
- $Child^*(x_0, y_0)$ . Checking if  $x_0$  is one of the ancestors of  $y_0$  is  $O(|E|)$  operations, by performing a depth-first search from  $x_0$ .
- $Child^+(x_0, y_0)$ . Same as the previous case, plus checking in constant time that  $x_0$  and  $y_0$  are not the same node.
- $Following(x_0, y_0)$ . Performing depth-first search on the DAG and checking if  $x_0$  is visited before  $y_0$ , but is not one of  $y_0$ 's ancestors costs  $O(|V| + |E|)$  operations.
- $NextSibling(x_0, y_0)$ . Checking if  $x_0$  is the previous sibling of  $y_0$  is also  $O(|E|)$ , by checking for each of the parents whether  $y_0$  is  $x_0$ 's next sibling relative to that parent.
- $NextSibling^+(x_0, y_0)$ . Same as the previous case, without the restriction that there is no other node after  $x_0$  and before  $y_0$ .
- $NextSibling^*(x_0, y_0)$ . Same as the previous case, but here  $x_0 = y_0$  may hold.

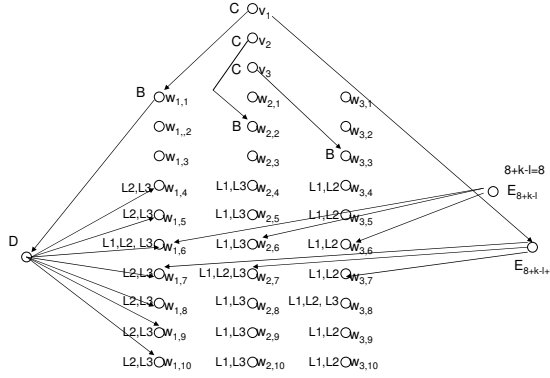
There is a satisfying assignment of nodes to variables iff then the non-deterministic Turing machine returns a positive answer in polynomial time. Therefore, the problem is in NP.  $\square$

#### NP-Hardness

**Theorem 3.** *Conjunctive queries over the signature  $\{Child^+(\cdot, \cdot), Label_a(\cdot)_{a \in \Sigma}\}$  are NP-Complete w.r.t. query complexity.*

*Proof.* We show a reduction from the NP-Complete problem of Boolean conjunctive queries over trees which use the signature  $\{Child^+(\cdot, \cdot), Child(\cdot, \cdot), Label_a(\cdot)_{a \in \Sigma}\}$  to the problem of Boolean conjunctive queries over DAGs which use the signature  $\{Child^+(\cdot, \cdot), Label_a(\cdot)_{a \in \Sigma}\}$ . We take the tree of Figure 2 and transform it to form the DAG in Figure 3. We also rewrite the query, polynomially, so it would use conjuncts from the signature  $\{Child^+(\cdot, \cdot), Label_a(\cdot)_{a \in \Sigma}\}$  in a way which ensures that the rewritten query is satisfied by the DAG iff the original query is satisfied by the tree which completes the proof.

Consider the DAG in Figure 3. It consists of all the nodes of the tree in Figure 2. All the original edges were deleted. Following is a list of nodes and



**Fig. 3.** DAG for proving NP-Hardness of  $\{Child^+(\cdot, \cdot), Label_a(\cdot)_{a \in \Sigma}\}$

edges that are added. Note that in order not to overload Figure 3, only samples of the new nodes and edges appear in it:

- An edge from every  $C$ -labeled node to the  $B$ -labeled node that used to be of distance three edges from the  $C$ -labeled node in the original tree.
- An edge from every  $B$ -labeled node to a new  $D$ -labeled node is added (three new  $D$ -labeled nodes are added, only one is shown in Figure 3).
- Edges from each  $D$ -labeled node  $d$  to all the  $L$ -labeled nodes that used to be descendants of  $d$ 's parent (a  $B$ -labeled node).
- Edges from every  $C$ -labeled node to a new node labeled by  $E_{8+k-l}$  where  $1 \leq k, l \leq 3$ , and from every  $E_{8+k-l}$ -labeled node to the three  $L$ -labeled nodes that used to be of distance  $8+k-l$  edges from the  $C$ -labeled node in the original tree.

The query for this DAG, denoted  $q'$ , is a modification of  $q$ , the query that was used for the tree, and is built as follows:

- For  $1 \leq i \leq m$ , the conjuncts  $C(x_i)$  and  $B(y_i)$  remain the same and  $Child^+(x_i, y_i)$  replaces  $Child^3(x_i, y_i)$ .
- The conjunct  $L_k(z_{k,l,i,j})$  remains the same.
- The conjuncts  $Child^+(y_i, d)$ ,  $D(d)$ ,  $Child^+(d, z_{k,l,i,j})$  replace  $Child^+(y_i, z_{k,l,i,j})$ .
- The conjuncts  $Child^+(x_j, e)$ ,  $E_{8+k-l}(e)$ ,  $Child^+(e, z_{k,l,i,j})$  replace  $Child^{8+k-l}(x_j, z_{k,l,i,j})$ .

Note that  $q'$  uses the  $Child^+(\cdot, \cdot)$  relation only, and is therefore compatible with the signature  $\{Child^+(\cdot, \cdot), Label_a(\cdot)_{a \in \Sigma}\}$ . The procedure of creating the query is polynomial, since every conjunct either stays the same or is replaced by a constant number of conjuncts. We argue that  $q$  is satisfied when evaluated on the original tree iff  $q'$  is satisfied when evaluated on the DAG.

- In the DAG every  $C$ -labeled node has exactly one  $B$ -labeled descendant. Evaluating  $q$  on the tree,  $Child^3(x_i, y_i)$  forces the bindings for  $\{x, y\}$  to

only be  $\{v_1, w_{1,1}\}$  or  $\{v_2, w_{2,2}\}$  or  $\{v_3, w_{3,3}\}$ . Evaluating  $q'$  on the DAG,  $Child^+(x_i, y_i)$  forces the same bindings. Since the conjuncts  $C(x_i)$  and  $B(y_i)$  remain the same, and none of the new nodes are  $B$  or  $C$  labeled,  $x_i$  and  $y_i$  have the exact same possible bindings in both cases.

- $z_{k,l,i,j}$  can only be bound to the one node which is both a descendant of distance  $8 + k - l$  (in edges) from  $x_j$  and a descendant of  $y_i$ , because of the conjuncts  $L_k(z_{k,l,i,j})$ ,  $Child^+(y_i, z_{k,l,i,j})$  and  $Child^{8+k-l}(x_j, z_{k,l,i,j})$  in  $q$ . In the DAG, the semantics of  $Child^+(y_i, z_{k,l,i,j})$  is achieved by  $Child^+(y_i, d)$ ,  $D(d)$ ,  $Child^+(d, z_{k,l,i,j})$  since nodes that used to be descendants of a  $B$ -labeled node in the tree ( $y_i$ ) are now children of the corresponding  $D$ -labeled node. The conjuncts  $Child^+(x_j, e)$ ,  $E_{8+k-l}(e)$ ,  $Child^+(e, z_{k,l,i,j})$  force that the only possibility for a binding for  $z_{k,l,i,j}$  would be the node which used to be of distance  $8+k-l$  edges distant  $x_j$ , exactly as it is with  $q$  applied to the tree.

This completes the reduction, which shows that the problem under discussion is NP-Hard. According to Theorem 2 the problem is in NP, therefore, the problem is NP-Complete.  $\square$

**Theorem 4.** *Conjunctive queries over the signature  $\{Child(\cdot, \cdot), Label_a(\cdot)_{a \in \Sigma}\}$  are NP-Complete w.r.t. query complexity.*

*Proof.* The proof for  $\{Child^+(\cdot, \cdot), Label_a(\cdot)_{a \in \Sigma}\}$  holds here as well, after replacing all  $Child^+(\cdot, \cdot)$  conjuncts with  $Child(\cdot, \cdot)$  conjuncts. The query forces that a satisfying binding for a  $Child^+(\cdot, \cdot)$  occurrence also satisfies  $Child(\cdot, \cdot)$ .  $\square$

**Theorem 5.** *Conjunctive queries over the signature  $\{Child^*(\cdot, \cdot), Label_a(\cdot)_{a \in \Sigma}\}$  are NP-Complete w.r.t. query complexity.*

*Proof.* The proof for  $\{Child^+(\cdot, \cdot), Label_a(\cdot)_{a \in \Sigma}\}$  holds here as well, after replacing all  $Child^+(\cdot, \cdot)$  conjuncts with  $Child^*(\cdot, \cdot)$  conjuncts, since the labels in the query force that  $Child^*(x_0, x_0)$  does not hold for any  $x_0$ .  $\square$

**Theorem 6.** *Conjunctive queries over the signature  $\{NextSibling(\cdot, \cdot), Label_a(\cdot)_{a \in \Sigma}\}$  are NP-Complete w.r.t. query complexity.*

*Proof.* We show a reduction from the NP-Complete problem of Boolean conjunctive queries over DAGs which use the signature  $\{Child^+(\cdot, \cdot), Label_a(\cdot)_{a \in \Sigma}\}$  to the problem of Boolean conjunctive queries over DAGs which use the signature  $\{NextSibling(\cdot, \cdot), Label_a(\cdot)_{a \in \Sigma}\}$ . We take the DAG described in Figure 3 and transform it to form the DAG in Figure 4. We also rewrite the query, polynomially, so it would use conjuncts from the signature  $\{NextSibling(\cdot, \cdot), Label_a(\cdot)_{a \in \Sigma}\}$  in a way which ensures that the rewritten query is satisfied by the DAG of Figure 4 iff the original query is satisfied by the DAG of Figure 3, similar to the way we proved in Theorem 3.

The DAG used in proving Theorem 3 (see Figure 3) is changed as follows. All the edges are deleted, and a new node is added for each deleted edge. Denote by  $n(u, v)$  the node added when deleting the edge  $(u, v, k)$ . The edges



query, delete  $L_k(z_{k,l,i,j})$ , and add  $G(z_{k,l,i,j}, L_k(p), Child^\bullet(z_{k,l,i,j}, p))$ , where for Theorem 5,  $\bullet$  is  $*$ , for Theorem 4,  $\bullet$  is  $+$ , and it is empty for Theorem 3.

For the proofs of Theorems 6, 7 and 8, eliminating multiple labels can be done by adding to every  $L_k$ -labeled node  $l$  up to three siblings, each through a new, different parent, and label  $l$  with  $G$ . In the query, delete  $L_k(z_{k,l,i,j})$ , and add  $G(z_{k,l,i,j}, L_k(p), NextSibling^\bullet(z_{k,l,i,j}, p))$ , where for Theorem 8,  $\bullet$  is  $*$ , for Theorem 7,  $\bullet$  is  $+$ , and it is empty for Theorem 6.

## 4 Conclusions

We characterize the complexity of conjunctive queries over DAGs, where query conjuncts are predicate corresponding to axis relations or label relations. Axis relations, which represent the DAG are a generalization of XPath axes for querying trees. We reduce the NP-Complete problem of evaluating Boolean conjunctive queries over trees over the signature  $\{Child^+(\cdot, \cdot), Child(\cdot, \cdot), Label(\cdot)\}$  to conjunctive query problems, with simple signatures, over DAGs. We prove that evaluation of queries over the signature  $\{Following(\cdot, \cdot), Label(\cdot)\}$ , which is polynomial for evaluation on trees, remains polynomial for evaluation over DAGs. We are currently extending this investigation to Datalog queries with built-in XPath predicates.

**Acknowledgment.** We thank Mukund Raghavachari for pointing us in this fruitful direction.

## References

1. Abiteboul, S., Hull, R., Vianu, V.: *Foundations of Databases*. Addison-Wesley (1995).
2. Adi, A., Etzion, O.: The situation manager rule language. RuleML (2002) 1–12.
3. Aho, A. V., Hopcroft, J. E. and Ullman, J. D.: *Design and Analysis of Computer Algorithms*. Addison Wesley (1974).
4. Antoniou, G., Van Harmelen, F.: *Semantic Web Primer*. MIT Press (2004) 3–4.
5. Garey, M., Johnson, D.: *Computers and intractability*. W. H. Freeman (1979) 259.
6. Gootlob, G., Schulz, K.: Conjunctive queries over trees. PODS (2004) 189–200.
7. Luckham, D. C., Frasca, B.: Complex event processing in distributed systems Computer systems laboratory technical report. CSL-TR-98-754. Stanford University (1998).
8. Vardi, Y. M.: The complexity of relational query languages. 14th ACM Symposium on Theory of Computing. New York (1982) 137–145.
9. W3C: RDF primer. <http://www.w3.org/TR/rdf-primer/>.
10. W3C: OWL- Web Ontology Language Overview. <http://www.w3.org/TR/owl-features/>.
11. XPath 1999. XML Path Language (XPath) version 1.0: <http://www.w3.org/TR/xpath> (1999).

# Incrementally Computing Ordered Answers of Acyclic Conjunctive Queries\*

Benny Kimelfeld and Yehoshua Sagiv

The Selim and Rachel Benin School of Engineering and Computer Science  
The Hebrew University of Jerusalem  
Edmond J. Safra Campus  
Jerusalem 91904, Israel  
{bennyk, sagiv}@cs.huji.ac.il

**Abstract.** Evaluations of SQL queries with the **ORDER BY** clause is considered. The naive approach of first computing the result and then sorting the tuples is not suitable for Web applications, since the result could be very large while users expect to get quickly the top- $k$  tuples. Tractability, in this case, amounts to enumerating answers in sorted order with *polynomial delay*, under *query-and-data* complexity. It is proved that an efficient algorithm for finding the top-ranked tuple of a conjunctive query is a sufficient (and not just necessary) condition for tractability. Several classes of orders are shown to have this property when queries are acyclic.

## 1 Introduction

Computing queries incrementally has become an increasingly important issue; for example, in Web applications users want to see the first few pages of results as quickly as possible. This task is more challenging if a ranking function is involved and users are interested in the *top-k* answers. One approach [3, 4] is to extend SQL with the **STOP AFTER** operator that limits the generated tuples to the top- $k$ . An efficient processing of this operator is done by enriching query plans with counters and sorters [3] and range partitioning [4]. Other papers proposed algorithms for computing the top- $k$  results of ranked joins, including the  $J^*$  algorithm [14], the rank-join algorithm [11] and the top- $k$ -oriented variant of the hash-join algorithm [9].

The goal of this paper is to investigate the complexity of computing answers of queries in ranked order. To discern between tractable and intractable cases, one has to measure the running time as a function of the combined size of the input and the output, using *query-and-data complexity* (that is, the size of the query is unbounded). Since queries may have exponentially many answers, an algorithm is efficient if it computes all the answers in *polynomial total time*, i.e., the runtime is bounded by a polynomial in the combined size of the query, the data and the result. Computing efficiently the top- $k$  answers requires a stronger notion, namely, the runtime should be polynomial in the size of the input and

---

\* This research was supported by The Israel Science Foundation (Grant 893/05).

the value  $k$ . An even stronger notion is that of *polynomial delay* [12], that is, the elapsed time between generating two successive tuples is polynomial only in the size of the input (i.e., the query and data).

The algorithms mentioned above are not efficient from a theoretical point of view, since the delay in producing the top answer (or any answer thereafter) could be exponential. We show that for a large class of queries, answers can be computed in ranked order with polynomial delay. We believe that this result could lead to better algorithms for computing the top- $k$  answers than those proposed thus far.

Two factors determine whether the top- $k$  answers can be computed efficiently. First, the *non-emptiness* problem must be tractable. In other words, if it is hard to determine whether the result of a query is nonempty, then it is impossible to enumerate the answers with polynomial delay (regardless of which order is used). Second, for some (rather simple) ranking functions, it is hard to enumerate the answers in ranked order even if non-emptiness is easy to check.

In this paper, we consider the class of *acyclic conjunctive queries*, which is one of the largest classes with an efficient algorithm for checking non-emptiness. We first show that, with respect to all conjunctive queries, the important property of ranking functions is the ability to find the top answer efficiently (i.e., in polynomial time in the size of the the query and the data). Clearly, this is a necessary condition for enumerating in ranked order with polynomial delay. In the case of general conjunctive queries, it is (rather surprisingly) also a sufficient condition. We identify several types of ranking functions that have this property with respect to acyclic conjunctive queries. Our results hold even if projections are used and duplicates should be removed, and even if the ranking depends also on attributes that are not included in the final projection. Note that a naive handling of duplicate elimination (i.e., removing them at the end of the computation) cannot yield an enumeration with polynomial delay or even in polynomial total time.

## 2 Motivation

For motivating our work, consider the database scheme of Figure 1 and the SQL query of Figure 2. This query finds apartments for sale and sorts the result according to a linear combination of the average temperature, the price and the distance from London. Note that the result includes only some of the attributes that determine the order; in other words, there are attributes that appear in the `ORDER BY` clause but not in the `SELECT` clause. But this fact does not affect the complexity of evaluating the given query, since duplicates are not removed (i.e., `DISTINCT` is not specified in the `SELECT` clause). That is, the difference between applying projection as soon as possible and applying it in the last step is negligible. If, however, duplicates have to be removed and projection is performed only in the last step (followed by duplicate elimination), then intermediate results could be much larger than the final one, leading to a difference of an exponential factor in the runtime.

|  |
|--|
| <ul style="list-style-type: none"> <li>– <i>Apartments</i> (id, city, street, price, bedrooms)</li> <li>– <i>Climate</i> (city, avgTemp)</li> <li>– <i>Distances</i> (fromCity, toCity, distance)</li> </ul> |
|--|

**Fig. 1.** A database scheme

|   |
|---|
| <pre> SELECT A.city, A.bedrooms, A.price FROM Apartments A, Climate C, Distances D WHERE A.city = C.city = D.fromCity AND D.toCity='London' ORDER BY C.avgTemp*10 - D.distance - A.price/1000 DESC </pre> |
|---|

**Fig. 2.** An SQL query

Two criteria determine whether queries can be evaluated without generating intermediate results that are too large. First, testing *non-emptiness* of queries (regardless of the order) must be a tractable problem. Since not all conjunctive queries satisfy this requirement [5], we state our actual tractability results for *acyclic conjunctive queries* [1, 2, 15] that have this property. Second, there must be an efficient algorithm for finding the top-ranked (i.e., first) tuple of the result, according to the specified order. Consider, for example, the following query.

```

SELECT *
FROM  $R_1, \dots, R_n$ 
ORDER BY  $\text{ABS}(R_1.A_1 + \dots + R_n.A_n - K)$ 

```

$R_1, \dots, R_n$  are names of relations that have the attributes  $A_1, \dots, A_n$ , respectively, and  $K$  is a number. **ABS** is the absolute-value operator. Testing non-emptiness of this query is easy, since it is merely a Cartesian product. It is intractable, however, to compute the top-ranked tuple.<sup>1</sup>

Our goal is to find sufficient conditions for efficiently enumerating all tuples of the result in sorted order. Informally, efficiency means that the delays between generating successive tuples are small. We show that for general conjunctive queries, tractability of finding the top-ranked tuple is a sufficient (and not just necessary) condition for efficiently enumerating in sorted order. This result holds even if duplicates are eliminated and attributes of the **ORDER BY** clause are not required to appear in the **SELECT** clause. The proof is by means of an algorithm that reduces the problem of enumerating in sorted order to the problem of finding the top-ranked tuple with respect to the given order. (Technically, we must find the top-ranked tuple with respect to the query that is obtained from the original one by adding all attributes to the **SELECT** clause.)

Next, we consider acyclic conjunctive queries and show that for two classes of orders, the top-ranked answer can be found efficiently. The first class consists

<sup>1</sup> This can be easily proved by a reduction from the *subset-sum* problem.



of *monotonic orders* that have the following property. Consider a tuple  $t$  and all subsets  $S$  of the attributes that appear in the **ORDER BY** clause. Informally, if for some subset  $S$ , replacing the values of  $S$  in  $t$  with new ones yields a tuple that has at least the same rank as  $t$ , then this replacement can only improve every tuple of the result that is equal to  $t$  on  $S$  (whenever the modified tuple is also in the result). As an example, the following orders are monotonic:

- **ORDER BY**  $C_1(R_{i_1}.A_{i_1}) + \dots + C_k(R_{i_k}.A_{i_k})$  [DESC] (**Linear combination**)
- **ORDER BY**  $R_{i_1}.A_{i_1}$  [DESC],  $\dots$ ,  $R_{i_k}.A_{i_k}$  [DESC] (**Lexicographic**)

The second class of orders is called *c-determined*, where  $c$  is a fixed positive integer (i.e.,  $c$  does not depend on the input). Informally, in a  $c$ -determined order, the position of a tuple in the order is determined by a set of  $c$  or fewer values. For example, the following two orders are 1-determined and 3-determined, respectively. Note that in the first order,  $k$  is unbounded (i.e., not assumed to be fixed).

- **ORDER BY**  $\text{MAX}(C_1 R_{i_1}.A_{i_1}, \dots, C_k R_{i_k}.A_{i_k})$  DESC
- **ORDER BY**  $R_1.A_1 * R_2.A_2 + R_3.A_3$

### 3 Formal Setting

Our goal is to identify queries and orders that have efficient algorithms for producing answers in the specified ordering. We consider *acyclic conjunctive queries*, since they satisfy the necessary requirement, namely, answers can be generated efficiently in an arbitrary order. This section defines the main concepts.

#### 3.1 Databases and Conjunctive Queries

We assume that **Dom** is a countable domain of atomic values. A *relation scheme*  $R$  has an associated *arity*, denoted by  $\text{arity}(R)$ . A *relation instance* (or simply, a *relation*) over a relation scheme  $R$  is a finite subset of **Dom** <sup>$\text{arity}(R)$</sup> . The elements of a relation are *tuples*. A *database scheme*  $S$  is a set of relation schemes. A *database instance* (or simply, a *database*)  $D$  over a database scheme  $S$  consists of a relation instance, denoted by  $D[R]$ , for each relation scheme  $R$  of  $S$ . We write  $D' \subseteq D$  if  $D'$  is obtained from  $D$  by deleting some tuples.

A *conjunctive query* (abbr. CQ)  $Q$  over a database scheme  $S$  has the form

$$Q(\mathbf{u}): - R_1(\mathbf{u}_1), R_2(\mathbf{u}_2), \dots, R_k(\mathbf{u}_k)$$

and it satisfies the following: (1)  $R_1, \dots, R_n$  are (not necessarily distinct) relation schemes of  $S$ ; (2)  $\mathbf{u}$  is a list of *variables*; (3) Each  $\mathbf{u}_i$  is a list of *terms* that has a length of  $\text{arity}(R_i)$ , where a term is either a constant of **Dom** or a variable; and (4) Each variable in  $\mathbf{u}$  appears in some  $\mathbf{u}_i$ . We say that each  $R_i(\mathbf{u}_i)$  is a *conjunct* of  $Q$  and  $Q(\mathbf{u})$  is the *head* of  $Q$ . The set of variables appearing in  $Q$  is denoted by  $\text{var}(Q)$ . When we consider a CQ  $Q$  and a database  $D$ , we implicitly assume that  $Q$  and  $D$  are defined over the same database scheme.

Consider a conjunctive query  $Q$  and a database  $D$ . An *assignment* for  $\text{var}(Q)$  is a mapping  $\varphi : \text{var}(Q) \rightarrow \mathbf{Dom}$ . We use  $\varphi \mathbf{u}_i$  to denote the tuple that is obtained from  $\mathbf{u}_i$  by replacing every variable  $X$  with  $\varphi(X)$  (and leaving constants unchanged). A *homomorphism from  $Q$  to  $D$*  is an assignment  $\varphi$  for  $\text{var}(Q)$ , such that  $\varphi \mathbf{u}_i \in D[R_i]$  for each conjunct  $R_i(\mathbf{u}_i)$  of  $Q$ . We use  $\text{Hom}(Q, D)$  to denote the set of all homomorphisms from  $Q$  to  $D$ . An *answer* is a tuple  $\varphi \mathbf{u}$ , where  $\varphi$  is a homomorphism of  $\text{Hom}(Q, D)$ . The *result* of applying  $Q$  to  $D$ , denoted by  $Q(D)$ , is the set of all the answers, i.e.,  $Q(D) = \{\varphi \mathbf{u} \mid \varphi \in \text{Hom}(Q, D)\}$ .

Acyclic conjunctive queries [1, 2] (abbr. ACQs) are conjunctive queries that have join trees. Formally, we say that a conjunctive query  $Q$  is *acyclic* if there exists a tree  $T$  (called a *join tree*), such the nodes of  $T$  are the conjuncts of  $Q$  and the following holds. For every variable  $X \in \text{var}(Q)$ , the subgraph of  $T$  that is induced by the conjuncts containing  $X$  is connected (i.e., it is a subtree of  $T$ ). It has been shown [15] that acyclic conjunctive queries can be computed efficiently, i.e., in time that is polynomial in the combined size of the query, the database and the result, while for general conjunctive queries, it is NP-complete just to determine whether the result is nonempty [5].

### 3.2 Orders over Query Answers

Consider a conjunctive query  $Q$  and a database  $D$ . Our goal is to enumerate all the tuples of  $Q(D)$  according to an order that is specified, for example, in the ORDER BY clause of an SQL query. Recall that the ORDER BY clause may include attributes that do not necessarily appear in the SELECT clause. In other words, we cannot assume that the specified order is defined merely over the tuples of the result, since these tuples are obtained after projecting out attributes that may determine the position of a tuple in the desired ordering. Thus, an order over the answers to a query should be defined in terms of a more general order on the *ways* to derive these answers, i.e., homomorphisms. As an example, consider the SQL query of Figure 2. The corresponding conjunctive query is

$$Q(C, B, P) :- \text{Apartments}(I, C, S, P, B), \text{Climate}(C, T), \\ \text{Distances}(C, \text{'London'}, D)$$

and the ranking function is  $10\varphi(T) - \varphi(D) - \varphi(P)/1000$ . The values  $\varphi(T)$  and  $\varphi(D)$ , however, do not appear in the result. Things become more complicated if the DISTINCT operator is used in order to eliminate duplicates. In this case, the position of a tuple is determined by the first appearance that this tuple would have, had the DISTINCT operator not been used.

Formally, we assume that there is an underlying *order*  $\succeq$  over homomorphisms that is defined for a given CQ  $Q$  and a database  $D$ . The order  $\succeq$  is a binary relation over  $\text{Hom}(Q, D)$  that is reflexive, transitive and total (i.e., every two homomorphisms are comparable).

Orders on homomorphisms determine orders on answers as follows. Consider a query with the head  $Q(\mathbf{u})$  and a database  $D$ . Let  $\varphi_1, \varphi_2 \in \text{Hom}(Q, D)$ . If  $\varphi_1 \succeq \varphi_2$  holds, then it means that  $\varphi_1$  generates a “better” answer, i.e.,  $\varphi_1 \mathbf{u}$

should precede  $\varphi_2 \mathbf{u}$ . Formally,  $\succeq$  implies an order  $\succeq_{Q,D}$  over the tuples of  $Q(D)$  that is defined as follow. Two tuples  $t_1, t_2 \in Q(D)$  satisfy  $t_1 \succeq_{Q,D} t_2$  if for every homomorphism  $\varphi_2 \in \text{Hom}(Q, D)$  satisfying  $\varphi_2 \mathbf{u} = t_2$ , there exists a homomorphism  $\varphi_1 \in \text{Hom}(Q, D)$  satisfying  $\varphi_1 \mathbf{u} = t_1$ , such that  $\varphi_1 \succeq \varphi_2$ . In other words,  $t_1 \succeq_{Q,D} t_2$  if the maximal homomorphisms  $\varphi_1$  and  $\varphi_2$  that produce  $t_1$  and  $t_2$ , respectively, satisfy  $\varphi_1 \succeq \varphi_2$ . Thus, the goal is to enumerate the answers of  $Q(D)$  in decreasing order, i.e., if  $t_1 \succeq_{Q,D} t_2$ , then  $t_1$  is printed before  $t_2$ .

### 3.3 Measuring Efficiency

We want to enumerate all the answers of  $Q(D)$  in the order  $\succeq_{Q,D}$ . A related problem is that of finding a maximal (i.e., top-ranked) homomorphism of  $\text{Hom}(Q, D)$ . An algorithm for this problem is deemed efficient if its runtime is polynomial (in the size of  $Q$  and  $D$ ). Note that we assume that  $\varphi' \succeq \varphi$  can be tested efficiently.

For the problem of enumerating all the answers of  $\text{Hom}(Q, D)$ , polynomial time is not a suitable measure of efficiency, since the output size could be much larger (e.g., exponentially larger) than the input size. For this type of problems, there are several notions of efficiency [12]. The strongest one among them is enumeration with *polynomial delay*, that is, after the algorithm prints the  $(i-1)$ st answer, it generates the next ( $i$ th) answer in time that is polynomial in the input size. Polynomial delay implies an efficient evaluation of the top- $k$  answers, since we can stop the execution after printing the first  $k$  answers and the runtime is only linear in  $k$  (and polynomial in the input).

By definition, the result of a conjunctive query is a set whereas an SQL query may produce duplicates (unless the `DISTINCT` operator is used). It is sufficient, however, to develop an efficient algorithm for enumerating tuples of the result assuming that duplicates are eliminated. If duplicates should be retained, then one can simply use this algorithm as if all the attributes appear in the `SELECT` clause (equivalently, as if all the variables of  $\text{var}(Q)$  appear in the head) and apply the required projection just before printing each tuple.

## 4 Ordered Evaluation with Polynomial Delay

### 4.1 The Main Theorem

In this section, we prove our main theorem. In the next section, we describe families of orders that satisfy the first condition of this theorem, assuming that the CQ  $Q$  is acyclic. We start with some notation and definitions.

Consider an ACQ  $Q$ , with the head  $Q(\mathbf{u})$ , and a database  $D$ . We denote by  $Q^+$  the query that is obtained from  $Q$  by adding the conjunct  $R_X(X)$  for each variable  $X$  of  $\mathbf{u}$ . Note that  $R_X$  is a relation scheme with arity 1. The database  $D^{Q(\mathbf{u})}$  is obtained from  $D$  by adding a relation instance  $I_X$  for each  $R_X$  as follows.  $I_X$  comprises all constants  $d$  of  $D$ , such that there is a conjunct  $R_i(\mathbf{u}_i)$  of  $Q$ , the variable  $X$  is in  $\mathbf{u}_i$  and the constant  $d$  appears in the relation  $R_i[D]$  in a column that corresponds to  $X$ . In other words, the relation for  $R_X$  contains (at least) all constants that could potentially be the image of  $X$  under some homomorphism from  $Q$  to  $D$ . The following proposition is rather straightforward.

**Proposition 1.** *Let  $Q$  be a conjunctive query and  $D$  be a database. Then,*

- $Q(D) = Q^+(D^{Q(\mathbf{u})})$ ;
- $\text{Hom}(Q, D) = \text{Hom}(Q^+, D^{Q(\mathbf{u})})$ ; and
- $Q$  is acyclic if and only if  $Q^+$  is acyclic.

According to the above proposition, if we are interested in evaluating a CQ  $Q$  over a database  $D$  or finding a maximal homomorphism of  $\text{Hom}(Q, D)$ , we can instead solve these problems with respect to the query  $Q^+$  and the database  $D^{Q(\mathbf{u})}$ . Furthermore, this transformation preserves acyclicity of  $Q$ , namely, if  $Q$  is a cyclic, then so is  $Q^+$ .

Our main result is the next theorem that uses the above proposition in order to show the following. An efficient algorithm for finding a maximal homomorphism is sufficient for efficiently enumerating all answers and all homomorphisms of a CQ  $Q$  with respect to a database  $D$ . Formally, we need to make the natural assumption that if an algorithm is efficient with respect to a CQ  $Q$  and a database  $D$ , then it is also correct and efficient with respect to every database  $\hat{D}$  that is obtained from  $D$  by deleting some tuples (i.e.,  $\hat{D} \subseteq D$ ). Note that when considering  $\hat{D}$  instead of  $D$ , we assume that the order on the homomorphisms of  $\text{Hom}(Q, \hat{D})$  is the same as the one defined on  $\text{Hom}(Q, D)$ .

**Theorem 1.** *Consider a CQ  $Q$ , a database  $D$  and an order  $\succeq$  on  $\text{Hom}(Q, D)$ . The following are equivalent.*

1. *There is a polynomial-time algorithm for solving the following problem. Given the CQ  $Q$  and a database  $\hat{D} \subseteq D^{Q(\mathbf{u})}$ , find a maximal homomorphism of  $\text{Hom}(Q^+, \hat{D})$ . (The running time is polynomial in the size of  $Q^+$  and  $\hat{D}$ .)*
2. *There are enumeration algorithms that run with polynomial delay for the following two problems. Given the CQ  $Q$  and a database  $\hat{D} \subseteq D^{Q(\mathbf{u})}$ , enumerate all the answers of  $Q^+(\hat{D})$  and all the homomorphisms of  $\text{Hom}(Q^+, \hat{D})$  in ranked order. (The delay is polynomial in the size of  $Q^+$  and  $\hat{D}$ .)*

Clearly, the second part implies the first. So, we now prove the other direction. Our proof is for the case where we want to enumerate  $Q(D)$  (or, equivalently,  $Q^+(D^{Q(\mathbf{u})})$ ). For that, we describe the algorithm  $\text{SORTEDEVAL}_{\succeq}(Q, D)$  of Figure 3 that enumerates the answers of  $Q(D)$  in ranked order, using the subroutine  $\text{MAXIMIZE}_{\succeq}(Q^+, \hat{D})$  that finds a maximal homomorphism of  $\text{Hom}(Q^+, \hat{D})$ , where  $\hat{D} \subseteq D^{Q(\mathbf{u})}$ . Our algorithm is an adaptation of a procedure<sup>2</sup> by Lawler [13] for computing the top- $k$  solutions to discrete optimization problems. Proving the theorem for an arbitrary  $\hat{D} \subseteq D^{Q(\mathbf{u})}$  requires straightforward modifications to the algorithm. Note that in order to use  $\text{SORTEDEVAL}_{\succeq}(Q, D)$  for enumerating all the homomorphisms of  $\text{Hom}(Q, D)$ , we need to modify  $Q$  so that the head includes all the variables of  $Q$ .

The algorithm  $\text{SORTEDEVAL}_{\succeq}(Q, D)$  uses two types of *constraints*. A *positive* constraint has the form  $X = a$  and a *negative* constraint has the form  $Y \neq b$ ,

<sup>2</sup> This procedure generalizes an algorithm of Yen [16] for enumerating simple paths.

```

SORTEDEVAL≥( $Q, D$ )
1   $\mathbf{u} \leftarrow$  the tuple of variables from the head of  $Q$ 
2   $Queue \leftarrow$  an empty priority queue, with priority based on  $\succeq$ 
3   $\varphi \leftarrow \text{MAXIMIZE}_{\succeq}(Q^+, D^{Q(\mathbf{u})})$ 
4  if  $\varphi \neq \perp$ 
5    then  $Queue.\text{INSERT}(\langle \emptyset, \emptyset, \varphi \rangle)$ 
6  while  $Queue$  is not empty
7    do  $\langle P, N, \varphi \rangle \leftarrow Queue.\text{REMOVETOP}()$ 
8       $\{X_1, \dots, X_k\} \leftarrow$  the variables that appear in  $\mathbf{u}$  but not in  $P$ 
9      for  $i \leftarrow 1$  to  $k$ 
10        do  $P_i \leftarrow P \cup \{X_1 = \varphi(X_1), \dots, X_{i-1} = \varphi(X_{i-1})\}$ 
11           $N_i \leftarrow N \cup \{X_i \neq \varphi(X_i)\}$ 
12           $D_i \leftarrow D^{Q(\mathbf{u})}$ 
13          for all  $(X = a) \in P_i$ 
14            do remove from  $D_i[R_X]$  all tuples except for  $\langle a \rangle$ 
15          for all  $(Y \neq b) \in N_i$ 
16            do remove from  $D_i[R_Y]$  the tuple  $\langle b \rangle$ 
17           $\varphi_i \leftarrow \text{MAXIMIZE}_{\succeq}(Q^+, D_i)$ 
18          if  $\varphi_i \neq \perp$ 
19            then  $Queue.\text{INSERT}(\langle P_i, N_i, \varphi_i \rangle)$ 
20    PRINT( $\varphi\mathbf{u}$ )

```

**Fig. 3.** Incrementally computing answers in sorted order

where both  $X$  and  $Y$  are variables from the head of  $Q$  while  $a$  and  $b$  are constants of  $D$ . We use  $P$  and  $P_i$  to denote sets of positive constraints whereas  $N$  and  $N_i$  denote sets of negative constraints. A homomorphism  $\varphi$  satisfies the sets of constraints  $P$  and  $N$  if  $\varphi(X) = a$  for all constraint  $X = a$  of  $P$  and  $\varphi(Y) \neq b$  for every constraint  $Y \neq b$  of  $N$ .

$\text{SORTEDEVAL}_{\succeq}$  uses a priority queue,  $Queue$ . The operations on  $Queue$  take logarithmic time in the size of  $Queue$  (hence, polynomial time in  $Q$  and  $D$ ). An element of  $Queue$  is a triplet  $\langle P, N, \varphi \rangle$ , where  $P$  and  $N$  are sets of positive and negative constraints, respectively, and  $\varphi \in \text{Hom}(Q, D)$  is a maximal homomorphism among all those satisfying  $P$  and  $N$ . Priority in  $Queue$  is based on  $\succeq$ ; that is, the top element of  $Queue$  is a triplet  $\langle P, N, \varphi \rangle$ , such that  $\varphi \succeq \varphi'$  for all triplets  $\langle P', N', \varphi' \rangle$  in  $Queue$ . A triplet  $\langle P, N, \varphi \rangle$  represents the subset of  $Q(D)$  comprising all answers that are produced by homomorphisms satisfying  $P$  and  $N$ , where  $\varphi$  is the “best” such homomorphism. The triplets of  $Queue$  represent disjoint subsets that cover all the answers that have not yet been printed.

The first element inserted into  $Queue$  (Lines 3–5) represents the whole set  $Q(D)$ . In the loop of Lines 7–20, each iteration starts by removing the top

element  $\langle P, N, \varphi \rangle$  from *Queue*. The answer  $\varphi \mathbf{u}$  is printed in Line 20. Let  $X_1, \dots, X_k$  be the variables that appear in  $\mathbf{u}$  but not in  $P$  (intuitively, these are the free variables of the answers represented by  $\langle P, N, \varphi \rangle$ ). In Lines 8–19, we partition all the answers represented by  $\langle P, N, \varphi \rangle$ , except for  $\varphi \mathbf{u}$ , into  $k$  disjoint subsets by creating the elements  $\langle P_i, N_i, \varphi_i \rangle$ ,  $1 \leq i \leq k$ , and inserting them into *Queue* (whenever  $\varphi_i$  exists, i.e.,  $\varphi_i \neq \perp$ ). The set  $P_i$  is obtained from  $P$  by adding the constraints  $X_j = \varphi(X_j)$  for  $j = 1, \dots, i-1$ . The set  $N_i$  is obtained from  $N$  by adding the constraint  $X_i \neq \varphi(X_i)$ . The maximal homomorphism  $\varphi_i$ , under  $P_i$  and  $N_i$ , is generated in Lines 13–17 by executing  $\text{MAXIMIZE}_{\succeq}(Q^+, D_i)$ , where  $D_i$  is the database that is obtained by changing  $D^{Q(\mathbf{u})}$  so that all constraints are satisfied: (1) For each constraint  $X = a$  of  $P_i$ , we remove from  $D^{Q(\mathbf{u})}[R_X]$  all tuples except for  $\langle a \rangle$ , and (2) for each constraint  $Y \neq b$  of  $N_i$ , we remove the tuple  $\langle b \rangle$  from  $D^{Q(\mathbf{u})}[R_Y]$ . The following lemma shows the correctness of the algorithm  $\text{SORTEDEVAL}_{\succeq}(Q, D)$ . It also shows that this algorithm enumerates with polynomial delay if  $\text{MAXIMIZE}_{\succeq}$  runs in polynomial time.

**Lemma 1.** *Given a CQ  $Q$ , the algorithm  $\text{SORTEDEVAL}_{\succeq}(Q, D)$  enumerates  $Q(D)$  in ranked order. The delay is polynomial provided that  $\text{MAXIMIZE}_{\succeq}$  runs in polynomial time.*

## 4.2 Tractable Orders

In this Section, we present two families of orders that satisfy the first part of Theorem 1 and, hence, also the second part, assuming that the CQ  $Q$  is acyclic.

**Monotonic Orders.** This family includes many widely used orders, such as sums of single-variable functions (or products, if the functions are positive) and lexicographic orders. In order to define monotonic orders, we need to consider partial assignments, i.e., mappings from some of the variables of the given query to constants of the database. First, consider two partial assignments  $\varphi$  and  $\varphi'$  that are defined on the sets of variables  $V$  and  $V'$ , respectively. The *combined* assignment  $\varphi' \oplus \varphi$  maps variables of  $V'$  according to  $\varphi'$  and variables of  $V \setminus V'$  according to  $\varphi$  (i.e.,  $\varphi'$  takes precedence when mapping variables that are in the domains of both assignments). Now, an order  $\succeq$  is *monotonic* if for all partial assignments  $\varphi, \varphi_1$  and  $\varphi_2$ , such that  $\varphi_1$  and  $\varphi_2$  are defined on the same set of variables, if  $\varphi_1 \succeq \varphi_2$  then  $\varphi_1 \oplus \varphi \succeq \varphi_2 \oplus \varphi$ .<sup>3</sup> As an example, suppose that each variable  $X$  is associated with a ranking function  $\text{rank}_X$ . Then, the following definitions of  $\text{rank}(\varphi)$  imply monotonic orders (regardless of the specific form of  $\text{rank}_X$ ). Note that  $V$  denotes the domain of  $\varphi$ .

$$\text{rank}(\varphi) = \sum_{X \in V} \text{rank}_X(\varphi(X)),$$

$$\text{rank}(\varphi) = \max\{\text{rank}_X(\varphi(X)) \mid X \in V\},$$

<sup>3</sup> For simplicity's sake, the definition of monotonic orders is more restrictive than necessary. A more general definition can be obtained by taking into account the structure of the join tree of the given query.

$$\text{rank}(\varphi) = \min\{\text{rank}_X(\varphi(X)) \mid X \in V\}.$$

The class of monotonic orders also contains the *lexicographic orders*. For example, every order specified by the SQL command `ORDER BY  $R_1.A_1, \dots, R_k.A_k$`  (with an optional `DESC` attached to each attribute) is lexicographic. We define this type of orders on homomorphisms.<sup>4</sup> Given a CQ  $Q$ , a lexicographic order  $\succeq$  is specified by a sequence  $(X_1, \dots, X_k)$  of distinct variables of  $Q$ . We assume that for every variable  $X_i$ , there is an order  $>_i$  over the values (of the given database  $D$ ) that  $X_i$  could be mapped to. Two homomorphisms  $\varphi_1, \varphi_2 \in \text{Hom}(Q, D)$  satisfy  $\varphi_1 \succeq \varphi_2$  if either (1)  $\varphi_1$  and  $\varphi_2$  agree on every variable  $X_i$  or (2) the minimal  $i$ , such that  $\varphi_1(X_i) \neq \varphi_2(X_i)$ , satisfies  $\varphi_1(X_i) >_i \varphi_2(X_i)$ .

We briefly sketch how to find a maximal homomorphism, under a monotonic order, for a given ACQ  $Q$  and a database  $D$ . Note that the running time is polynomial in the size of  $Q$  and  $D$ . First, we construct a rooted join tree  $T$  of  $Q$  (recall that the nodes of  $T$  are the conjuncts of  $Q$ ). Then, we traverse  $T$  bottom up. For each conjunct  $R(\mathbf{u}_i)$  and tuple  $t \in D[R]$ , we compute the maximal partial assignment  $\varphi_t$  to the variables of the subtree rooted at  $R(\mathbf{u}_i)$ , such that  $\varphi_t$  maps  $\mathbf{u}_i$  to  $t$ . A detailed description is beyond the scope of this paper.

**c-determined Orders.** This type of orders generalizes the one given in [6] and is defined as follows. Given a fixed positive integer  $c$ , an order  $\succeq$  over homomorphisms is *c-determined* if for all  $\varphi \in \text{Hom}(Q, D)$ , there exists a set  $C_\varphi$  of  $c$  or fewer variables, such that every homomorphism  $\varphi' \in \text{Hom}(Q, D)$  that agrees with  $\varphi$  on the variables of  $C_\varphi$  must satisfy  $\varphi' \succeq \varphi$ . For example, an order that is defined by a ranking function over 3 variables, e.g., the one used in the query of Figure 2, is 3-determined. As another example, an order that is defined by taking the maximal rank over all variables (as defined above) is 1-determined. Note that a  $c$ -determined order is not necessarily monotonic and vice versa (even if the order is determined by a fixed set of variables  $C$ , i.e.,  $C_\varphi = C$  for all  $\varphi$ ).

Given an ACQ  $Q$ , a database  $D$  and a  $c$ -determined order  $\succeq$ , we can find a maximal homomorphism as follows. We consider every subset  $C$  of  $\text{var}(Q)$ , with  $c$  or fewer variables, and every possible assignment  $\psi$  that maps  $C$  to constants from  $D$ . For each  $\psi$ , we compute a homomorphism  $\varphi_C$  (if it exists) that agrees with  $\psi$  on  $C$ . The desired homomorphism is the maximal one among all the  $\varphi_C$ .

The following theorem summarizes the above discussion.

**Theorem 2.** *Consider an ACQ  $Q$  and database  $D$ . For the following orders, finding a maximal homomorphism is polynomial in  $Q$  and  $D$ . Hence, all answers can be generated in ranked order with polynomial delay.*

- Monotonic (e.g., lexicographic) orders.
- $c$ -determined orders, where  $c$  is a fixed positive integer.

<sup>4</sup> We omit both the definition of these orders on partial assignments and a proof that they are monotonic, due to a lack of space.

## 5 Space-Efficient Evaluation

The algorithm `SortedEval` of Figure 3 runs with polynomial delay, but its queue may grow linearly with respect to the number of answers. In this section, we briefly discuss an alternative technique that applies to certain types of orders. This technique has a polynomial delay and uses only a polynomial amount of space. It is based on the following proposition.

**Proposition 2.** *Let  $Q$  be an ACQ and  $D$  be a database. The answers of  $Q(D)$  can be enumerated (in an arbitrary order) with polynomial delay and polynomial space.*

If the order is either lexicographic or  $c$ -determined by a fixed set of variables and, moreover, it depends only on variables that appear in the head of  $Q$ , then we can use the above result as follows. We bind the variables of the head that determine the rank to constants from the database. Then, we sort the different bindings. In the case of a  $c$ -determined order, there is only a polynomial number of bindings. If the order is lexicographic, then we have to do it one variable at a time, in a recursive manner. For a given binding, we can determine whether there is any answer and, if so, enumerate all the answers using the above proposition. Additional details are beyond the scope of this paper.

## 6 Conclusion

We have investigated the complexity of incrementally computing SQL queries with the `ORDER BY` clause. We have stated actual tractability results for SQL queries that correspond to acyclic conjunctive queries, since solving efficiently the non-emptiness problem is a necessary condition for incremental computation with polynomial delay. In our framework, the `ORDER BY` clause is modeled as an order on homomorphisms from the conjunctive query to the database, since the `ORDER BY` clause may include attributes that do not appear in the `SELECT` clause.

Our main result holds for general conjunctive queries and it states that if one can find the top-ranked homomorphism in polynomial time, then all the tuples of the result can be enumerated in sorted order with polynomial delay (subject to the natural assumption that the given algorithm for finding a maximal homomorphism can be applied efficiently and correctly to any database that is obtained from the given one by deleting some tuples). For acyclic conjunctive queries, we have shown that  $c$ -determined and monotonic (which include lexicographic) orders satisfy this property. If all attributes of the `ORDER BY` clause also appear in the `SELECT` clause, then the computation requires only polynomial space if the order is either  $c$ -determined by a fixed set of variables  $C$  or lexicographic. Our results can be easily extended to unions of acyclic conjunctive queries. In this case, enumeration in sorted order runs in incremental polynomial time if duplicates are eliminated; otherwise, the delay is polynomial.

Some related work dealt with problems that are similar to, yet different from the one considered in this paper. In [7, 8], the goal is to find the top- $k$  objects,



where ranking is determined by a monotone function of several attributes and the ranking order on each attribute is given by a sorted list. The PREFER system [10] is aimed at finding the top- $k$  tuples of a relation, based on a combination of several ranking functions. Query processing is optimized by using sorted views.

In summary, the worst-case delay of our algorithms is much better (polynomial vs. exponential) than the delays of previous algorithms, but more work is needed in order to incorporate our algorithms in practical systems.

## References

1. C. Beeri, R. Fagin, D. Maier, and M. Yannakakis. On the desirability of acyclic database schemes. *J. ACM*, 30(3):479–513, 1983.
2. P. A. Bernstein and N. Goodman. Power of natural semijoins. *SIAM J. Comput.*, 10(4):751–771, 1981.
3. M. J. Carey and D. Kossmann. On saying "enough already!" in SQL. In *SIGMOD*, pages 219–230, 1997.
4. M. J. Carey and D. Kossmann. Reducing the braking distance of an SQL query engine. In *VLDB*, pages 158–169, 1998.
5. A. K. Chandra and P. M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *STOC*, pages 77–90, 1977.
6. S. Cohen and Y. Sagiv. An incremental algorithm for computing ranked full disjunctions. In *PODS*, 2005.
7. R. Fagin. Combining fuzzy information from multiple systems. *J. Comput. Syst. Sci.*, 58(1):83–99, 1999.
8. R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. *J. Comput. Syst. Sci.*, 66(4):614–656, 2003.
9. D. Habich, W. Lehner, and A. Hinneburg. Optimizing multiple top- $k$  queries over joins. In *SSDBM*, pages 195–204, 2005.
10. V. Hristidis, N. Koudas, and Y. Papakonstantinou. PREFER: A system for the efficient execution of multi-parametric ranked queries. In *SIGMOD*, 2001.
11. I. F. Ilyas, W. G. Aref, and A. K. Elmagarmid. Supporting top- $k$  join queries in relational databases. In *VLDB*, pages 754–765, 2003.
12. D. S. Johnson, M. Yannakakis, and C. H. Papadimitriou. On generating all maximal independent sets. *Information Processing Letters*, 27:119–123, March 1988.
13. E. L. Lawler. A procedure for computing the  $k$  best solutions to discrete optimization problems and its application to the shortest path problem. *Management Science*, 18:401–405, 1972.
14. A. Natsev, Y. C. Chang, J. R. Smith, C. S. Li, and J. S. Vitter. Supporting incremental join queries on ranked inputs. In *VLDB*, pages 281–290, 2001.
15. M. Yannakakis. Algorithms for acyclic database schemes. In *VLDB*, pages 82–94, 1981.
16. J. Y. Yen. Another algorithm for finding the  $k$  shortest loopless network paths. In *"Proc. 41st Mtg. Operations Research Society of America"*, volume 20, page B/185, 1972.

# Count-Constraints for Generating XML

Sara Cohen

Technion—Israel Institute of Technology  
sarac@ie.technion.ac.il

**Abstract.** The problem of automatically generating XML documents that satisfy given user constraints is considered. Specifically, given (1) a count-constraint  $C$ , expressed using XPath and the aggregate function `count` and (2) a DTD  $D$ , the problem of generating an XML document that satisfies both  $C$  and  $D$  is studied. Classes of count-constraints and DTDs for which the satisfiability problem is polynomial are identified. For these classes it is shown that a satisfying document can be generated efficiently in terms of the input and the output.

## 1 Introduction

XML is generally perceived as an important component for solving a variety of real-world problems, and is used extensively in data processing applications. Many researchers have studied properties of XML and proposed algorithms for XML processing. However, when testing an XML application or algorithm, an annoying problem often crops up. Although XML use seems to be widespread, it is frequently difficult to find or generate appropriate XML documents for testing purposes.

Currently, there are three distinct sources for XML documents to be used for testing. First, there are several *downloadable XML datasets* that can be found on the web. Popular examples include Shakespeare plays,<sup>1</sup> the CIA factbook,<sup>2</sup> and the DBLP.<sup>3</sup> These sources are often used for experimentation in academic papers. *XML benchmarks* [9, 4, 10, 8, 5] are a second source for XML documents. These benchmarks generally supply both data and queries, and are often geared for testing implementations of XQuery. Both downloadable XML datasets and XML benchmarks are not always suitable for experimentation and analysis of a particular application or algorithm. Specifically, these sources may not even be legal input for a given application, since they may not conform to some required schema. Hence, they may not be usable at all. Even if they are legal input, they may not have certain characteristics that the given algorithm specializes in processing, e.g., deep documents, documents with large branching factors, etc.

*XML generators* [1, 2] are a third source for XML. An XML generator is a program that generates XML according to some given user constraints. XML generators provide the user with flexibility in defining the XML documents of

---

<sup>1</sup> Available at <http://www.ibiblio.org/xml/examples/shakespeare/>

<sup>2</sup> Available at <http://www.dbis.informatik.uni-goettingen.de/Mondial/>

<sup>3</sup> Available at <http://dblp.uni-trier.de/xml/>

choice. However, their usefulness is greatly determined by the level of expressibility allowed for the users to define the desired target documents. In [1], both the structure and content of the output documents are randomly generated. As input, the user may provide parameters that control the randomness of the result, e.g., the number of levels in the resulting tree and the minimum and maximum number of children for a given level. Clearly, with this system it is not possible to generate a document that conforms to some desired schema. ToXgene [2] allows a finer level of control when generating documents. In ToXgene, the user defines the desired output by providing an XML schema annotated with constraints that specify different characteristics of the target documents, such as the minimum and maximum number of children for a specific element. Much of the focus in ToXgene is in allowing the user to specify which sources should be used to generate the actual data in the documents.

Although ToXgene allows the user to specify the target schema of the documents generated, the types of structural constraints that can be specified are rather limited. In particular, in ToXgene, the constraints are specified *locally* within an XML Schema. Hence, the constraints must refer to *specific elements* in *specific locations* of the schema. *Global* constraints on the resulting desired documents cannot be expressed. For example, in ToXgene one may require there to be at least 100 *b* children of an *a* element. However, one cannot require that there are at least 100 *b* elements in the entire document. This constraint is not expressible, since it does not refer to a specific location of *b*. One also cannot require that there are at least 100 elements of any type in the document. This constraint is also not expressible since it does not refer to a specific element.

In this paper we consider the problem of building an XML generator that allows for global constraints over the result documents. Our constraints are expressible in terms of XPath and the aggregation function `count`. Specifically, we assume that the user provides a DTD *D* and a count-constraint *C*. Our goal is to efficiently generate a document satisfying both *C* and *D*, if such a document exists. Note that we only concern ourselves with creating a satisfying document that has the correct structure. Defining the content (i.e., the actual string values within the elements) of such satisfying documents is also important, but is beyond the scope of the current paper.

Our contributions can be summarized as follows.

- We start by presenting a language for specifying global constraints over XML documents (Section 2). The language proposed combines XPath expressions with the aggregate function `count` to yield a natural, yet powerful, constraint language. We also define the problems of interest in this paper. Informally, given a count-constraint *C* and a DTD *D*, we are interested in two problems. The first problem is to determine whether a document exists that satisfies both *C* and *D*. The second problem is to generate such a satisfying document, if one exists.
- We show, for several classes of count-constraints and DTDs, that the satisfiability problem is intractable (Section 3). This result is important since

it implies that for these classes of count-constraints and DTDs, an efficient XML generator cannot be implemented.

- For the other important classes of count-constraints and DTDs, we show that the satisfiability problem is in PTIME (Section 4). In addition, we show that satisfying documents can be generated in polynomial time with respect to the input and output.

## 2 Framework

In this section we formally define count-constraints, DTDs and our satisfiability problem. These definitions are an adaption of similar definitions appearing in previous works which study satisfiability of XPath expressions, especially [3].

### 2.1 Count-Constraints

We consider count-constraints that are written using standard XPath expressions and the function `count`. We start by formally defining the types of constraints considered.

An *XPath expression*  $p$  has the form

$$p ::= \epsilon \mid l \mid \downarrow \mid \downarrow^* \mid p/p \mid p[\text{count}(p)\theta k],$$

where  $l$  is a label,  $\theta$  is one of  $\leq, \geq$ , and  $k$  is a nonnegative number. Intuitively,  $\downarrow$  is the child axis,  $\downarrow^*$  is the descendent-or-self axis and  $/$  is the concatenation operator.

A *count-constraint*  $C$  has the form

$$C ::= \text{count}(p)\theta k, \mid (C \wedge C) \mid (C \vee C),$$

where  $p$  is an XPath expression, and as before,  $\theta$  is one of  $\leq, \geq$  and  $k$  is a nonnegative number.<sup>4</sup> We say that a count-constraint is *simple* if it does not contain any logical operator ( $\wedge$  or  $\vee$ ). We will usually use  $c$  to denote simple count-constraints and  $C$  to denote count-constraints that are not necessarily simple.

In the sequel, we will refer to the class of count-constraints defined above as  $\mathcal{C}(\downarrow, \downarrow^*, [\text{count}], \leq, \geq, \wedge, \vee)$ . We will sometimes also be interested in restricted classes of count-constraints. We denote such classes by listing the operators allowed, e.g.,  $\mathcal{C}(\downarrow^*, [\text{count}], \geq, \wedge)$  is the class that contains count-constraints that

- may use the axis  $\downarrow^*$  (but not  $\downarrow$ ),
- allow nested qualifications expressed using  $[]$  and `count`,
- have the comparison operator  $\geq$  (but not  $\leq$ )
- and may have the logical operator  $\wedge$  (but not  $\vee$ ).

<sup>4</sup> It is sufficient to consider the comparison operators  $\leq$  and  $\geq$ . Clearly,  $=$  can be expressed using  $\leq$  and  $\geq$ . In addition,  $<, >$  are expressible since  $k$  is an integer, and thus, for example,  $\text{count}(p) < k \equiv \text{count}(p) \leq k - 1$ .

To ensure some minimal expressibility, the concatenation operator  $/$ , at least one of the operators  $\downarrow$  and  $\downarrow^*$ , and labels  $l$ , are allowed in all classes of count-constraints. Also, all our count-constraints use the **count** operator in the outermost level. (Classes without **[count]** do not use nested qualifications.) Therefore, all classes that we will consider will contain at least one of the operators  $\leq, \geq$ .

## 2.2 Satisfaction of a Count-Constraint

XML documents are modeled as trees, in the standard fashion. We will use  $T$  to denote XML trees, and  $n, n'$ , etc., to denote nodes. To simplify the presentation, our XML trees have only elements (and not attributes).

We define when a count-constraint is satisfied by an XML tree. For this purpose, we first consider satisfaction of XPath expressions by nodes in a tree. Let  $T$  be an XML tree, let  $n$  be a node in  $T$  and let  $p$  be an XPath expression. We say that  $T$  *satisfies*  $p$  at  $n$  if and only if there is a node  $n'$  such that  $T \models p(n, n')$ , defined inductively in terms of the structure of  $p$ , as follows:

1. if  $p = \epsilon$ , then  $n = n'$ ;
2. if  $p = l$ , then  $n'$  is a child of  $n$  and is labeled  $l$ ;
3. if  $p = \downarrow$ , then  $n'$  is any child of  $n$ ;
4. if  $p = \downarrow^*$ , then  $n'$  is either  $n$ , or is a descendent of  $n$ ;
5. if  $p = p_1/p_2$ , then there is a node  $n''$  such that  $T \models p_1(n, n'')$  and  $T \models p_2(n'', n')$ ;
6. if  $p = p_1[\text{count}(p_2) \theta k]$ , then  $T \models p_1(n, n')$  and  $\text{CntSat}(n', p_2) \theta k$ , where  $\text{CntSat}(n', p_2)$  is the number of nodes  $n''$  such that  $T \models p_2(n', n'')$ . Formally,  $\text{CntSat}(n', p_2) = |\{n'' \mid T \models p_2(n', n'')\}|$ .

Now, let  $r$  be the root of  $T$ . Let  $C$  be a count-constraint. We define when  $T$  *satisfies*  $c$ , written  $T \models C$ , as follows

1. if  $C = \text{count}(p) \theta k$ , then  $\text{CntSat}(r, p) \theta k$ ;
2. if  $C = C_1 \wedge C_2$ , then  $T \models C_1$  and  $T \models C_2$ ;
3. if  $C = C_1 \vee C_2$ , then  $T \models C_1$  or  $T \models C_2$ .

*Example 2.1.* Consider the following simple count-constraints.

$$\begin{aligned}
 c_1 &= \text{count}(\downarrow^*) \geq 1000 & c_4 &= \text{count}(\downarrow^*/a) \leq 10 \\
 c_2 &= \text{count}(\downarrow^*) \leq 999 & c_5 &= \text{count}(\downarrow^*/a/b) \geq 21 \\
 c_3 &= \text{count}(\downarrow^* [\text{count}(\downarrow) \geq 3]) \leq 0
 \end{aligned}$$

Intuitively,  $c_1$  states that there are at least 1000 nodes in the tree,  $c_2$  states that there are at most 999 nodes in the tree,  $c_3$  states that there is no node with more than two children (i.e., the branching factor is at most 2),  $c_4$  states that there are at most 10  $a$  nodes and  $c_5$  states that there are at least 21  $b$  children of  $a$  elements.

The constraints  $c_1$  and  $c_6$  are in the class  $\mathcal{C}(\downarrow^*, \geq)$  (and in any class containing this class). The constraints  $c_2$  and  $c_4$  are in the class  $\mathcal{C}(\downarrow^*, \leq)$ . The constraint  $c_3$  is in the class  $\mathcal{C}(\downarrow, \downarrow^*, [\text{count}], \leq, \geq)$ . Note that  $c_3$  has a nested **count** qualification.

Clearly, there are count-constraints that cannot be satisfied by any XML tree. An obvious example is  $c_1 \wedge c_2$ . Another, less immediate, example is  $c_3 \wedge c_4 \wedge c_5$ . This expression is not satisfiable it requires that a satisfying document contain at most 20 children of any type of  $a$  elements (because of the number of  $a$  elements and the branching factor), and also contain at least 21  $b$  children of  $a$  elements.

### 2.3 DTDs

To simplify the presentation, we consider DTDs without attribute definitions. A DTD  $D$  is a set of labels  $\mathcal{L}$ , each of which is associated with an *element description*. Formally, an element description has the form

$$e ::= \mathcal{S} \mid l \mid e, e \mid e|e \mid e? \mid e^*$$

We use  $\mathcal{S}$  to denote a string value, i.e., PCDATA, and  $l$  to denote a label in  $\mathcal{L}$ . Each DTD is associated with a root label, which we will denote  $r$ . Given a label  $l$ , we will use  $e(l)$  to denote the element description of  $l$ .

Satisfaction of an XML tree  $T$  with respect to a DTD  $D$  is defined in the obvious way, i.e., (1) the root of  $T$  must be labeled with  $r$ , (2) all the nodes in  $T$  must be labeled with values in  $\mathcal{L}$  and (3) the children of each node must satisfy the element description of its label.

There are DTDs for which no finite satisfying documents exist. For example, this is the case if the DTD contains the element description  $r \leftarrow r$ . Without loss of generality, we will not consider such DTDs in this paper. We note that such unsatisfiable DTDs can be recognized in linear time [3].

We use  $\mathcal{D}$  to denote a class of DTDs. In particular, we will consider the class of disjunction-free DTDs  $\mathcal{D}_{df}$  (which do not use the operator  $|$ ) and the class of nonrecursive DTDs  $\mathcal{D}_{nr}$  (for which all conforming documents have bounded depth). Finally, we will consider the class  $\mathcal{D}_{df,nr} = \mathcal{D}_{df} \cap \mathcal{D}_{nr}$  which contains DTDs which are both disjunction free and nonrecursive.

*Example 2.2.* As an example, a DTD  $D_1$ , for the labels  $\mathcal{L} = \{r, a, b, c, d\}$ , appears below.

$$\begin{array}{ll} r \leftarrow a, b^*, (a, b, c)? & c \leftarrow \mathcal{S} \\ a \leftarrow b, c, d? & d \leftarrow \mathcal{S} \\ b \leftarrow c, d & \end{array}$$

Observe that  $D_1$  is both disjunction-free and nonrecursive, i.e.,  $D_1 \in \mathcal{D}_{df,nr}$ . If we adapt the element description of  $b$  to contain  $a$ , then  $D_1$  would be recursive.

### 2.4 Problems of Interest

Given a tree  $T$ , a DTD  $D$  and a count-constraint  $C$ , we write  $T \models (C, D)$  if  $T$  satisfies both  $C$  and  $D$ . Let  $\mathcal{C}$  be a class of count-constraints and  $\mathcal{D}$  be a class of DTDs. In this paper we will be interested in the following *satisfiability problem*:

**SAT( $\mathcal{C}, \mathcal{D}$ ):** Given a DTD  $D \in \mathcal{D}$  and a count-constraint  $C \in \mathcal{C}$ , does there exist an XML tree  $T$  such that  $T \models (C, D)$ ?

In general, solving this problem is not easy, due to the complex interactions between the requirements of  $C$  and those implied by  $D$ .

Recall that our end goal is to develop an *efficient* system that receives as input both a DTD  $D$  and a count-constraint  $C$ , and returns a document that satisfies both  $D$  and  $C$ , or else outputs that such a document does not exist. Formally, our aim is to use our results on the satisfiability problem to solve the following additional problem.

**GEN( $\mathcal{C}, \mathcal{D}$ ):** Given a DTD  $D \in \mathcal{D}$  and a count-constraint  $C \in \mathcal{C}$ , generate an XML tree  $T$  such that  $T \models (C, D)$ , if such a document exists. Otherwise, output that the pair  $(C, D)$  are unsatisfiable.

We note that the output of GEN( $\mathcal{C}, \mathcal{D}$ ) may be exponentially larger than the input. To see why this holds, consider the count-constraint  $c_1$  from Example 2.1. Any document satisfying  $c_1$  will be exponentially larger than  $c_1$ , since it will contain at least 1000 nodes (and  $c_1$  uses the number 1000 in binary notation). Therefore, we will use input-output complexity as our yardstick of efficiency. In other words, our aim is to develop a system that returns a document that satisfies both the DTD and the count-constraint in polynomial time in the size of the input and the output.

Clearly, if the SAT( $\mathcal{C}, \mathcal{D}$ ) problem is intractable, then GEN( $\mathcal{C}, \mathcal{D}$ ) cannot be solved in polynomial time in the size of the input and output. Hence, our system will not be able to run efficiently. Therefore, we start by identifying classes  $\mathcal{C}$  and  $\mathcal{D}$  for which the SAT( $\mathcal{C}, \mathcal{D}$ ) problem is intractable (Section 3). Afterwards, we present classes for which SAT( $\mathcal{C}, \mathcal{D}$ ) is polynomial and GEN( $\mathcal{C}, \mathcal{D}$ ) can be solved in polynomial time in the input and the output (Section 4).

### 3 Intractable Classes

In this section we prove that the SAT( $\mathcal{C}, \mathcal{D}$ ) problem is intractable, for various classes of count-constraints and DTDs.

The reader may wonder if previous results on satisfiability of XPath expressions, e.g., [7, 6, 3], can be applied to prove intractability of classes of count-constraints. In answer to this question, we first note that no previous work considered XPath expressions containing the function `count`. Adding this function, even on the outermost level, makes our classes strictly more expressive than corresponding classes that do not use `count`. Formally, let  $\mathcal{X}$  be a class of XPath expressions (possibly different from those considered in our paper). Let  $\mathcal{C}_{\mathcal{X}}$  be the class of count-constraints which allows exactly the expressions `count( $p$ )  $\theta$   $k$` , for any  $p \in \mathcal{X}$ . Clearly, the SAT( $\mathcal{C}_{\mathcal{X}}, \mathcal{D}$ ) problem is at least as difficult as the corresponding SAT( $\mathcal{X}, \mathcal{D}$ ) problem, since  $p$  is satisfiable with respect to a tree  $T$  if and only if `count( $p$ )  $\geq 1$`  is satisfiable with respect to  $T$ . We note in passing that this reasoning was exactly our motivation for considering in this paper only a restricted set of XPath operators (e.g., restricted XPath axes) in our count-constraints. We ruled out operators for which previous complexity results implied that our corresponding satisfiability problem is intractable.

We now consider the problem at hand and present cases in which the  $\text{SAT}(\mathcal{C}, \mathcal{D})$  problem can be shown to be intractable. None of these results follow directly from previous complexity results.

Theorem 3.1 lists cases in which  $\text{SAT}(\mathcal{C}, \mathcal{D})$  is NP-hard. Note that allowing qualifications  $[\text{count}]$  is strictly more expressive than using  $\wedge$ , since any count-constraint  $c_1 \wedge c_2 \wedge \dots \wedge c_n$  can be written equivalently as  $\epsilon[c_1][c_2] \dots [c_n]$ . Hence, any result for a class containing  $\wedge$  also holds if we replace  $\wedge$  with  $[\text{count}]$ . Theorem 3.1 is proven by showing reductions from 3SAT. Different reductions are required for the different classes considered.

**Theorem 3.1.** *The following problems are NP-hard:*

1.  $\text{SAT}(\mathcal{C}(\downarrow^*, \geq, \wedge), \mathcal{D}_{nr});$
2.  $\text{SAT}(\mathcal{C}(\downarrow, \geq, \wedge), \mathcal{D}_{nr});$
3.  $\text{SAT}(\mathcal{C}(\downarrow^*, \leq, \geq, \wedge), \mathcal{D}_{df, nr});$
4.  $\text{SAT}(\mathcal{C}(\downarrow, \leq, \geq, \wedge), \mathcal{D}_{df, nr});$
5.  $\text{SAT}(\mathcal{C}(\downarrow^*, [\text{count}], \leq), \mathcal{D}_{df, nr});$
6.  $\text{SAT}(\mathcal{C}(\downarrow, [\text{count}], \leq), \mathcal{D}_{df, nr});$
7.  $\text{SAT}(\mathcal{C}(\downarrow^*, \leq, \wedge), \mathcal{D}_{nr});$
8.  $\text{SAT}(\mathcal{C}(\downarrow, \leq, \wedge), \mathcal{D}_{nr});$

## 4 Polynomial Classes

In this section we show for which classes of count-constraints and DTDs the satisfiability problem is in PTIME. From Cases 3 and 4 of Theorem 3.1, we can conclude that there is no tractable class which allows for conjunctions of count-constraints and has both  $\leq$ ,  $\geq$ . Hence, we start by considering the satisfiability problem for each of  $\leq$  and  $\geq$  separately. Then, we use these results to present a class of count-constraints that has both  $\leq$ ,  $\geq$ , and is also tractable.

### 4.1 Count-Constraints with “ $\geq$ ”

As a result of Cases 1 and 2 of Theorem 3.1, we must restrict the class of DTDs considered. Therefore, in this section we consider only the class of DTDs  $\mathcal{D}_{df}$ . We show that for this class of DTDs, and for count-constraints containing only  $\geq$ , the satisfiability problem is always solvable in PTIME. In other words, we prove that  $\text{SAT}(\mathcal{C}(\downarrow, \downarrow^*, [\text{count}], \geq, \wedge, \vee), \mathcal{D}_{df})$  is satisfiable in PTIME.

We start by showing that it is sufficient to consider simple count-constraints, that do not contain any disjunctions or conjunctions.

**Lemma 4.1.**  *$\text{SAT}(\mathcal{C}(\downarrow, \downarrow^*, [\text{count}], \geq, \wedge, \vee), \mathcal{D}_{df})$  is solvable in PTIME if and only if  $\text{SAT}(\mathcal{C}(\downarrow, \downarrow^*, [\text{count}], \geq), \mathcal{D}_{df})$  is solvable in PTIME.*

We now show that  $\text{SAT}(\mathcal{C}(\downarrow, \downarrow^*, [\text{count}], \geq), \mathcal{D}_{df})$  is solvable in PTIME. To do this, we define a function that counts the number of possible occurrences of a path rooted at a given label.



Let  $D$  be a DTD,  $p$  be an XPath expression, and  $l, l'$  be labels in  $\mathcal{L}$ . Let  $D_l$  be the DTD that is identical to  $D$ , but is rooted at  $l$ , instead of at the original root  $r$ . (Note that  $D_r = D$ .) Let  $\mathcal{T}_l$ , denote the (possibly infinite) set of all trees  $T$  that satisfy  $D_l$ . We define two functions,  $PCnt_D(T, p, l')$  and  $MaxPCnt_D(l, p, l')$ , as follows.

- Given a  $T \in \mathcal{T}_l$  rooted at a node  $n$ , we define  $PCnt_D(T, p, l')$  as the number of nodes  $n'$  in  $T$  such that
  1.  $p \models (n, n')$  and
  2.  $n'$  is labeled with  $l'$ .
- We define  $MaxPCnt_D(l, p, l')$  as the maximal value of  $PCnt_D(T, p, l')$ , for any tree  $T$  in  $\mathcal{T}_l$ . Formally,

$$MaxPCnt_D(l, p, l') = \begin{cases} \max\{PCnt_D(T, p, l') \mid T \in \mathcal{T}_l\} & \text{if a maximum exists} \\ \infty & \text{otherwise} \end{cases}$$

Intuitively,  $MaxPCnt_D(l, p, l')$  is the maximal number of nodes labeled with  $l'$  reachable via the path  $p$  from a node labeled with  $l$ . To make the presentation clearer, we omit the subscript  $D$  when the DTD is clear from the context.

*Example 4.2.* Recall the DTD  $D_1$  from Example 2.2. Consider the paths

$$p_1 = \downarrow^*/a \qquad p_2 = \downarrow^*/b \qquad p_3 = \downarrow^*/c$$

Observe that  $MaxPCnt(r, p_1, a) = 2$ , since a node labeled with  $r$  may have at most two descendants labeled with  $a$ . Observe also that  $MaxPCnt(r, p_2, b) = \infty$ , since there exist trees conforming to  $D_1$  with an arbitrary number of  $b$ -labeled descendants of an  $r$ -labeled node. Finally, observe that  $MaxPCnt(r, p_3, c) = \infty$ , while  $MaxPCnt(a, p_3, c) = 2$ , since there may be an infinite number of  $c$ -labeled descendants of a tree rooted at  $r$ , but there can only be two  $c$ -labeled descendant of a tree rooted at  $a$  (one as a child of  $a$ , and one as a grandchild via  $b$ ).

Note also that  $MaxPCnt(r, p_1, b) = 0$  and  $MaxPCnt(r, p_2, a) = 0$ . Intuitively, this holds since  $p_1$  considers  $a$ -labeled descendants of  $r$ , and none of these are labeled with  $b$ . Similarly,  $p_2$  considers  $b$ -labeled descendants of  $r$ , and none of these are labeled with  $a$ .

The above definition does not yield an efficient algorithm to compute the function  $MaxPCnt(l, p, l')$ , since infinitely many trees must be considered. However, the following theorem states that this function can, in fact, be efficiently computed.

**Theorem 4.3.** *Let  $D$  be a DTD in  $\mathcal{D}_{df}$ , let  $p$  be a path expression and let  $l$  and  $l'$  be labels. Then,  $MaxPCnt(l, p, l')$  can be computed in polynomial time in the size of  $D$  and  $p$ .*

*Proof (Sketch).* We start by considering nonrecursive DTDs. For such DTDs, we calculate  $MaxPCnt(l, p, l')$  by repeatedly solving smaller problems. To this end, we consider both the format of  $e(l)$  (the element description of  $l$ ) and the format

**Table 1.** A table showing how to compute  $MaxPCnt(l, p, l')$  for DTDs in  $\mathcal{D}_{df, nr}$ 

| $p$                             | $MaxPCnt(l, p, l')$   |
|---------------------------------|---|
| $\epsilon$                      | $\chi_{l'}(l)$  |
| $\downarrow^*$                  | 1, if $l = l'$ . $\delta(e(l), p, l')$ , otherwise.   |
| $\downarrow$                    | $\delta(e(l), \epsilon, l')$  |
| $l_1$                           | $\chi_{l'}(l_1) \times \delta(e(l), \epsilon, l')$  |
| $p_1/p_2$                       | $\sum_{l'' \in \mathcal{C}} (MaxPCnt(l, p_1, l'') \times MaxPCnt(l'', p_2, l'))$                |
| $p_1[\text{count}(p_2) \geq k]$ | $MaxPCnt(l, p_1, l')$ , if $\exists l''$ s.t. $MaxPCnt(l', p_2, l'') \geq k$ ;<br>0, otherwise. |

| $e$           | $\delta(e, p, l')$                        |
|---------------|---|
| $\mathcal{S}$ | 0   |
| $e_1, e_2$    | $\delta(e_1, p, l') + \delta(e_2, p, l')$ |
| $e_1?$        | $\delta(e_1, p, l')$                      |
| $e_1^*$       | $\infty \times \delta(e_1, p, l')$        |
| $l_1$         | $MaxPCnt(l_1, p, l')$                     |

of  $p$ . Table 1 summarizes how  $MaxPCnt(l, p, l')$  can be computed by analyzing the structure of  $p$  and  $e(l)$ . Note that in Table 1 we use  $\chi_{l'}$  to denote the characteristic function of  $l'$ , i.e.,  $\chi_{l'}(l)$  equals one if  $l' = l$  and equals zero, otherwise. The computation of  $MaxPCnt$  requires a call to a auxiliary function  $\delta$ , for path expressions  $\downarrow^*$ ,  $\downarrow$  and  $l$ .

It is easy to see that computing the function as described in Table 1 can be performed in polynomial time. It remains to be shown that this computation yields the correct values. We omit this proof due to lack of space. (In Example 4.4 we demonstrate the computation of  $MaxPCnt$ .)

Finally, we note that it is possible to compute  $MaxPCnt$  for recursive DTDs in a similar manner. Care has to be taken to avoid an infinite loop of computation, which would result if the computation was done as described in Table 1. However, this can easily be overcome by first analyzing which labels are recursive and then carefully applying an adapted calculation of Table 1.  $\square$

*Example 4.4.* We demonstrate the calculation of  $MaxPCnt(l, p, l')$  by considering the DTD  $D_1$  from Example 2.2 and the path  $p_3 = \downarrow^*/c$  from Example 4.2.

We show how to compute  $MaxPCnt(a, p_3, c)$  using Table 1. First, we apply the rule for paths of the form  $p/p$  and derive that

$$MaxPCnt(a, p_3, c) = \sum_{l \in \{r, a, b, c, d\}} MaxPCnt(a, \downarrow^*, l) \times MaxPCnt(l, c, c).$$

Therefore, to compute the value of  $MaxPCnt(a, p_3, c)$ , we must consider each label  $l$  and compute the values of  $MaxPCnt(a, \downarrow^*, l)$  and  $MaxPCnt(l, c, c)$ .

1.  $l = c$ : Observe that  $MaxPCnt(c, c, c) = \chi_c(c) \times \delta(\mathcal{S}, \epsilon, c)$ . Since  $\delta(\mathcal{S}, \epsilon, c) = 0$ , choosing  $l$  as  $c$  will only add 0 to the summation.
2.  $l = d$ : Using the same reasoning as in Case 1, we derive that 0 will be added to the summation.
3.  $l = b$ : We first calculate  $MaxPCnt(a, \downarrow^*, b)$ . Since  $e(a) = b, c, d?$ , we have that

$$\begin{aligned}
 MaxPCnt(a, \downarrow^*, b) &= \delta((b, c, d?), \downarrow^*, b) \\
 &= \delta(b, \downarrow^*, b) + \delta(c, \downarrow^*, b) + \delta(d?, \downarrow^*, b) \\
 &= \delta(b, \downarrow^*, b) + \delta(c, \downarrow^*, b) + \delta(d, \downarrow^*, b) \\
 &= MaxPCnt(b, \downarrow^*, b) + MaxPCnt(c, \downarrow^*, b) + MaxPCnt(d, \downarrow^*, b) \\
 &= 1 + \delta(\mathcal{S}, \downarrow^*, b) + \delta(\mathcal{S}, \downarrow^*, b) \\
 &= 1
 \end{aligned}$$

Now, we calculate  $MaxPCnt(b, c, c)$ . Recall that  $e(b) = c, d$ . Therefore,

$$\begin{aligned}
 MaxPCnt(b, c, c) &= \chi_c(c) \times \delta((c, d), \epsilon, c) \\
 &= \delta(c, \epsilon, c) + \delta(d, \epsilon, c) \\
 &= MaxPCnt(c, \epsilon, c) + MaxPCnt(d, \epsilon, c) \\
 &= \chi_c(c) + \chi_c(d) \\
 &= 1.
 \end{aligned}$$

Thus, in total, choosing  $l = b$  adds 1 to our summation.

4.  $l = a$ : Using similar reasoning as in Case 3, we derive that 1 will be added to the summation.
5.  $l = r$ : In this case, we will add 0 to the summation since it can be shown that  $MaxPCnt(r, c, c) = 0$ .

To conclude, we derive that  $MaxPCnt(a, \downarrow^*, c) = 2$ , as expected.

We can now prove the following results.

**Theorem 4.5.** *The  $SAT(\mathcal{C}(\downarrow, \downarrow^*, [\text{count}], \geq, \wedge, \vee), \mathcal{D}_{df})$  problem is solvable in PTIME. The  $GEN(\mathcal{C}(\downarrow, \downarrow^*, [\text{count}], \geq, \wedge, \vee), \mathcal{D}_{df})$  problem is solvable in polynomial time in the size of the input and the output.*

## 4.2 Count-Constraints with “ $\leq$ ”

We now consider count-constraints that use the comparison operator  $\leq$ . By careful consideration of Cases 5–8 of Theorem 3.1, one may conclude that the largest classes of count-constraints and DTDs for which the satisfiability problem

might be tractable are  $\mathcal{C}(\downarrow, \downarrow^*, \leq, \wedge, \vee)$  and  $\mathcal{D}_{df}$ , respectively. Indeed, we show that the  $\text{SAT}(\mathcal{C}(\downarrow, \downarrow^*, \leq, \wedge, \vee), \mathcal{D}_{df})$  problem is solvable in PTIME.

We start by observing that any DTD can be minimized to remove any optional parts. Formally, let  $D$  be a DTD. Let  $D^{min}$  be the DTD derived from  $D$  by removing all elements surrounded by a  $?$  or  $*$  sign. The root of  $D^{min}$  is the same as that of  $D$ .

*Example 4.6.* Recall the DTD  $D_1$  from Example 2.2. Then,  $D_1^{min}$  is the DTD defined by

$$\begin{array}{ll} r \leftarrow a & c \leftarrow \mathcal{S} \\ a \leftarrow b, c & d \leftarrow \mathcal{S} \\ b \leftarrow c, d & \end{array}$$

The DTD  $D^{min}$  is completely deterministic if  $D$  is disjunction-free. In other words, there is exactly one tree  $T^{min}$  that satisfies  $D^{min}$ . It is possible to show the following result.

**Proposition 4.7.** *Let  $C$  be a count-constraint in  $\mathcal{C}(\downarrow, \downarrow^*, \leq, \wedge, \vee)$  and let  $D$  be a DTD in  $\mathcal{D}_{df}$ . Then, there exists a tree  $T$  such that  $T \models (C, D)$  if and only if  $T^{min} \models (C, D)$ .*

Note that this proposition would no longer hold if nested [count] qualifiers were allowed in  $C$ .

It follows from Proposition 4.7 that in order to check whether the pair  $(C, D)$  is satisfiable, it is sufficient to generate the single tree  $T^{min}$  that satisfies  $D^{min}$ , and to check for satisfiability with respect to this tree. However, this approach does not yield PTIME evaluation, since the tree  $T^{min}$  may be exponential in the size of  $D$ . Instead, we must check whether  $C$  is satisfiable with respect to  $T^{min}$ , without creating  $T^{min}$ .

**Theorem 4.8.** *The  $\text{SAT}(\mathcal{C}(\downarrow, \downarrow^*, \leq, \wedge, \vee), \mathcal{D}_{df})$  problem is solvable in PTIME.*

Corollary 4.9 immediately follows from Theorem 4.8, since  $T^{min}$  is such a satisfying document.

**Corollary 4.9.** *The  $\text{GEN}(\mathcal{C}(\downarrow, \downarrow^*, \leq, \wedge, \vee), \mathcal{D}_{df})$  problem is solvable in polynomial time in the size of the input and the output.*

### 4.3 Count-Constraints with “ $\geq$ ” and “ $\leq$ ”

Using the results of the previous sections, we can show the following satisfiability result.

**Theorem 4.10.** *The  $\text{SAT}(\mathcal{C}(\downarrow, \downarrow^*, \leq, \geq, \vee), \mathcal{D}_{df})$  problem is solvable in PTIME and the  $\text{GEN}(\mathcal{C}(\downarrow, \downarrow^*, \leq, \geq, \vee), \mathcal{D}_{df})$  problem is solvable in polynomial time in the size of the input and output.*

## 5 Conclusion

In this paper, we studied the satisfiability problem for various classes of count-constraints and DTDs. We identified important classes for which this problem is tractable. These results are useful as a starting point for creating an XML generator that can deal with global constraints over the satisfying documents.

As future work we intend to implement our algorithms, thereby creating an efficient XML generator. We also are considering adding additional types of constraints (such as the ability to constrain the depth of the document). We will study the satisfiability problem in these new languages. Finally, we will consider the problem of finding maximal and minimal-sized satisfying documents, for a given DTD and count-constraint.

## References

1. A. Aboulmaga, J. Naughton, and C. Zhang. Generating synthetic complex-structured xml data. In *Proc. 5th International Workshop on the Web and Databases*, Madison (Wisconsin, USA), May 2001.
2. D. Barbosa, A. Mendelzon, J. Keenleyside, and K. Lyons. ToXgene: an extensible template-based data generator for xml. In *Proc. 4th International Workshop on the Web and Databases*, Santa Barbara (California, USA), June 2002.
3. M. Benedikt, W. Fan, and F. Geerts. XPath satisfiability in the presence of dtds. In *Proc. 24th Symposium on Principles of Database Systems*, Baltimore (Maryland, USA), June 2005. ACM Press.
4. T. Böhme and E. Rahm. XMach-1: A benchmark for xml data management. In *Datenbanksysteme in Büro, Technik und Wissenschaft*, Oldenburg (Germany), 2001.
5. S. Bressan, G. Dobbie, Z. Lacroix, M. Lee, Y. Li, U. Nambiar, and B. Wadhwa. X007: applying 007 benchmark to xml query processing tool. In *Proceedings of the 2001 ACM CIKM International Conference on Information and Knowledge Management*, Atlanta (Georgia, USA), Nov. 2001.
6. F. Geerts and W. Fan. Satisfiability of XPath queries with sibling axes. In *10th International Workshop on Database Programming Languages*, Trondheim (Norway), Aug. 2005. Springer-Verlag.
7. J. Hidders. Satisfiability of XPath expressions. In *9th International Workshop on Database Programming Languages*, pages 21–36, Potsdam (Germany), Sept. 2003. Springer-Verlag.
8. K. Runapongsa, J. Patel, H. Jagadish, Y. Chen, and S. Al-Khalifa. The Michigan benchmark: towards xml query performance diagnostics. *Information Systems*, 31(2):73–97, 206.
9. A. Schmidt, F. Waas, M. Kersten, M. Carey, I. Manolescu, and R. Busse. XMark: a benchmark for xml data management. In *Proc. 28th International Conference on Very Large Data Bases*, Hong Kong (China), Sept. 2002. Morgan Kaufmann Publishers.
10. B. Yao, M. Özsu, and N. Khandelwal. XBench benchmark and performance testing of xml dbms. In *Proc. 20th International Conference on Data Engineering*, Boston (Massachusetts, USA), Mar. 2004. IEEE Computer Society.

# A Data Model for Music Information Retrieval

Tamar Berman

Graduate School of Library and Information Science  
University of Illinois at Urbana-Champaign  
Champaign, IL 61820 USA  
tamar@uiuc.edu

**Abstract.** This paper describes a data model for the representation of tonal music. In this model, music is conceived as an equally-spaced time series of 12-dimensional vectors. The model has been successfully applied to the task of discovering frequently recurring patterns, and to the related task of retrieving user-defined musical patterns. This was accomplished by converting midi sequences of music by W.A. Mozart into the time series representation and analyzing these with data mining tools and SQL queries. The novelty of the pattern extraction capability supported by the model is in the potentially complex description of the sequences, which may contain both melodic and harmonic features, may be embedded within each other, or interspersed with other patterns or occurrences. A unique feature of the model is the use of time intervals as the basic representational unit, which fosters possibilities for future application to audio data.

## 1 Introduction

Musical patterns are recurring musical structures whose description may include pitch, time, dynamics, harmony, timbre, contour and motion. Musical pattern extraction is used in music generation, retrieval and analysis [1]. A considerable body of research exists in this area: Huron's Humdrum toolkit [2] can be used to perform searches using regular expressions; Meek and Birmingham [3] developed a tool for extracting recurring themes from collections of MIDI files; Downie and Nelson [4] and Doraisamy and Ruger [5] propose methods for retrieving sequences using the n-gram method, Birmingham et al. [6] base their retrieval on Markov model representation, McNab et al. [7] propose a method for sequence retrieval based on string matching.

The motivating task in most of these methods is matching a melody which is sung or played by a user with similar sequences stored in a database. Much of the challenge facing these systems is related to the imprecision of the query, as the singer or player may provide the system with incorrect or imprecise pitch or rhythm information.

The novelty of the pattern extraction method and the underlying representation described in this paper lie in their support for discovery and retrieval of frequently recurring, complex patterns, containing both melodic and harmonic features, which may be overlaid upon each other, embedded within each other or interspersed with data that is not part of the sequence. The *schemas*, described in paragraph 1.1, are patterns of this quality and were used as a test case for the performance of the system.

Another unique feature of the model's pattern extraction capability is the description of the distances between the musical events in the sequence in absolute time units. This time-based approach creates a common basis for analyzing pieces of different tempi, and opens possibilities for future application to audio data.

## 1.1 Musical Schemas

A schema is a term used by psychologists to describe an abstract mental representation of past experiences, which we use in evaluating new experiences [8]. A typical schema consists of objects and events that are related to each other in some way, each of which is characterized by several component features.

Meyer [9] describes musical schemas that were used by composers throughout the classical period. These schemas are comprised of two or more musical events that follow each other, each of which consists of a harmonic as well as a melodic element. For example, the 1-7...4-3 schema, described in detail by Gjerdingen [10], consists of two events:

1. The melody descends from the 1st degree to the 7th. The harmony shifts from I to V while the bass moves from 1 or 3 to 2, or descends from 1 to 5.
2. The melody descends from the 4th degree to the 3rd. The harmony shifts from V to I while the bass moves from 5 or 7 to 1.

These events follow each other within a time frame of a few seconds. Schema occurrences often overlap or are embedded within other schemas and patterns.

Composers in the Classic era were expected to be familiar with the schema phrases and fluent in their use. "Native listeners" of 18th-century court music were able to recognize these gestures and to appreciate the sophistication of their use by the masters [11].

## 1.2 The Model

This paragraph will describe the data representation model underlying the pattern extractor. In this model, music is conceived as an equally-spaced time series of 12-dimensional vectors.

A time series is a set of ordered observations on the value of one or more variables taken at successive points in time. The observations may be equally spaced in time.

In the data model described in this paper, each element in this time series, which we call a harmonic window, describes the pitch content of the time interval contained in the window. For example, the harmonic window which starts at 5 seconds into the piece and ends at 6 seconds into the piece will describe, for each pitch class, whether it is present or absent during the 6th second of the piece, and what role it plays within that time interval.

Therefore, the elements in the time series are 12-dimensional vectors with values from the set  $\{0,1,2,3\}$ , where 0 indicates absence, 2 indicates presence as bass, 3 indicates presence as top voice and 1 indicates presence which is neither top nor bottom.

For example, for a window length of 1 second and onset interval of 0.5 seconds, the note records shown in Table 1 translate into the harmonic windows of the time series shown in Table 2.

**Table 1.** Note records in the database

| Start Time | Duration | Pitch Class | Octave | Channel |
|------------|----------|-------------|--------|---------|
| 1.250      | 0.461    | G           | 2      | 1       |
| 1.719      | 0.148    | E           | 2      | 1       |
| 1.875      | 0.930    | C           | 3      | 1       |
| 2.500      | 0.617    | G           | 1      | 2       |
| 2.500      | 0.617    | E           | 1      | 2       |
| 2.812      | 0.062    | D           | 3      | 1       |
| 2.891      | 0.062    | C           | 3      | 1       |
| 2.969      | 0.062    | B           | 2      | 1       |

**Table 2.** Harmonic windows in the time series

| Start Time | C | C# | D | D# | E | F | F# | G | G# | A | A# | B |
|------------|---|----|---|----|---|---|----|---|----|---|----|---|
| 0.5        | 0 | 0  | 0 | 0  | 0 | 0 | 0  | 3 | 0  | 0 | 0  | 0 |
| 1.0        | 3 | 0  | 0 | 0  | 2 | 0 | 0  | 1 | 0  | 0 | 0  | 0 |
| 1.5        | 3 | 0  | 0 | 0  | 2 | 0 | 0  | 1 | 0  | 0 | 0  | 0 |
| 2.0        | 1 | 0  | 3 | 0  | 2 | 0 | 0  | 1 | 0  | 0 | 0  | 1 |

Table 2 shows that only G was present in the window [0.5, 1.5], the windows [1, 2] and [1.5, 2.5] contained C, E and G with C being the top voice and E being the bass, and the window [2, 3] contained C, D, E, G and B, with D being on top and E acting as bass. For windows containing a single pitch - such as [0.5, 1.5] in this example - it is tagged as the top voice.

The parameters *window length* and *onset interval* require some explanation: The onset interval defines the time that elapses between window onsets and is somewhat analogous to the sampling rate used in audio files, as it defines how often a “harmonic sample” is taken and recorded as a harmonic window. The window length describes the amount of tolerance permitted for pitches to be considered as sounding simultaneously. A very small window implies that pitches must be played together, or almost together, in order to be in the same window, whereas a wide window allows for successively sounding pitches to be grouped together. The harmonic window could be viewed as loosely analogous to the FFT window in audio data analysis, as it describes pitch presence in a given time frame.

Harmonic windows will overlap when the onset interval is smaller than the window length. When the onset interval and window length are equal to each other harmonic windows will not overlap, but will reflect all the harmonic information contained in the



midi file. When the onset interval is greater than the window size information loss will occur, as some harmonic information may not be recorded in any window.

It should be noted that, prior to transforming the music into the time series representation, the midi files must be transposed into a common initial key, so all begin in C major or A minor, and modulations that occur in each piece are transposed to be relative to this “anchor” scale. For example, a modulation to the dominant key in a major piece would be transposed to G major. Consequently, a finding such as “C&E&G is a frequent combination” can be read as “the tonic triad of the notated key is a frequent combination”. Similarly, a finding such as “G&B&D is frequent” actually means that the dominant triad of the notated key, or the tonic triad of the dominant key, is frequent.

### 1.3 Test Data

The test data consisted of 505 midi files of music by W.A. Mozart. These were obtained from the Classical Music Archives [12] and included symphonies, piano sonatas, piano concertos, other concertos and piano trios. To efficiently utilize computational resources, only the first 50 measures of each piece were included in the database.

## 2 Data Mining Experiment

The purpose of the data mining experiment was to examine the ability of a general-purpose data mining algorithm to detect frequently-recurring musical sequences in a music database. Specifically, could a mining algorithm discover sequences that support the 1-7...4-3 schema described above?

The algorithm for sequence discovery proposed by Agrawal and Srikant [13] discovers frequently occurring sequences in databases. The classic case described is that of the supermarket owner who wishes to learn whether products in the supermarket tend to be purchased in a particular order. For example, a purchase pattern could be: “A+B ...(purchases unrelated to the pattern) ... C+D (more unrelated purchases) F”, where ‘+’ indicates that the items were purchased in a single transaction.

The *support* of a sequence is defined as the fraction of total customers who support the sequence. A customer supports a sequence if it is contained in the sequence of all transactions made by that customer, ordered by transaction time. The problem of mining sequential patterns is to find the maximal sequences among all sequences that have a user-specified minimum support. Each such maximal sequence represents a sequential pattern. The *confidence* of a sequence is the probability of the consequent event occurring given that the antecedent events have occurred.

### 2.1 Example

Table 3 shows an example of customer transactions. The customer sequences derived from these are shown in Table 4. For minimum support = 25% (i.e. 2 customers), the maximal supported sequences are: <(30) (90)> and <(30) (40 70)>.

**Table 3.** Customer transactions example

| Customer | Transaction Time | Items Purchased |
|----------|------------------|-----------------|
| 1        | 6/25/03          | 30              |
| 1        | 6/30/03          | 90              |
| 2        | 6/10/03          | 10, 20          |
| 2        | 6/15/03          | 30              |
| 2        | 6/20/03          | 40, 60, 70      |
| 3        | 6/25/03          | 30, 50, 70      |
| 4        | 6/25/03          | 30              |
| 4        | 6/30/03          | 40, 70          |
| 4        | 7/25/03          | 90              |
| 5        | 6/12/03          | 90              |

**Table 4.** Customer sequences

| Customer | Customer Sequence         |
|----------|---------------------------|
| 1        | <(30) (90)>               |
| 2        | <(10 20) (30) (40 60 70)> |
| 3        | <(30 50 70)>              |
| 4        | <(30) (40 70) (90)>       |
| 5        | <(90)>                    |

This algorithm was chosen for the task of identifying musical sequences for the following reasons:

1. It is able to identify frequently recurring sequences in large databases.
2. It allows for each event in the sequence to be comprised of one or more items. This fits the description of the schemas as being comprised of events which contain a combination of pitches.
3. It is capable of identifying a sequence of events even if these events are separated by, or occur simultaneously with other patterns or occurrences that are not part of the sequence. This is an important quality of the schemas.

## 2.2 Results

Results for the data mining experiment are shown in table 5. These were obtained using an off-the-shelf analysis package [14] that implements the aforementioned algorithm, with the following parameters:

1. Maximum number of events in sequence: 4
2. Maximum duration of sequence in seconds: 5 seconds
3. Maximum number of pitches in each event: 3
4. Minimum support required of a pitch combination: 98% of pieces
5. Minimum support required of a sequence: 40% of pieces

These parameters were selected to optimize utilization of memory resources while preserving the constraints laid out in the schema description.

The results show a fair degree of support for the harmony of the 1-7...4-3 schema, in both its major and minor form. The major form (rows 1-4) showed support of 54%-59%, whereas the minor form (rows 5-7) showed support of 44%-47%. Clearly evident is the capability of the time series representation model to support this type of analysis.

**Table 5.** Results of the data mining experiment

|   | <b>Window Length (sec)</b> | <b>Onset Interval (sec)</b> | <b>Sequence</b>                | <b>Support</b> | <b>Confidence</b> |
|---|----------------------------|-----------------------------|--------------------------------|----------------|-------------------|
| 1 | 1.0                        | 0.50                        | C&E&G=>G&B&D<br>=>B&F=>C&E     | 59.60 %        | 61.55 %           |
| 2 | 1.0                        | 0.50                        | C&E&G=>G&B&D<br>=>B&D&F=>C&E&G | 54.26 %        | 56.38 %           |
| 3 | 0.5                        | 0.25                        | C&E&G=>G&B&D<br>=>G&B&F=>C&E&G | 56.24 %        | 58.32 %           |
| 4 | 0.5                        | 0.25                        | C&E&G=>G&B&D<br>=>G&D&F=>C&E&G | 55.64 %        | 58.06 %           |
| 5 | 1.0                        | 0.50                        | A&C&E=>E&G&B<br>=>D&G=>A&C&E   | 45.94 %        | 47.35 %           |
| 6 | 0.5                        | 0.25                        | A&C&E=>E&G&B<br>=>D&E=>A&C&E   | 47.33 %        | 48.98 %           |
| 7 | 0.5                        | 0.25                        | A&C&E=>E&G&B<br>=>B&D&E=>A&C&E | 44.55 %        | 46.30 %           |

### 3 Sequence Retrieval

The model supports retrieval of instances of a sequence based on parametric specification. The following parameters are currently supported:

1. Number of events in the sequence
2. Pitch combinations which must be present in each event
3. Role specification for each pitch within the event (Top/Bass/Middle/None)
4. Maximum duration of sequence in seconds
5. Maximum/minimum distance between neighboring events in seconds
6. Designation of a specific channel for a pitch, or specification that a melodic line present in the sequence be played along a single channel
7. Window length/lengths for the time series
8. Onset interval/intervals for the time series

The query returns the names of midi files containing the specified sequence, and pointers to the locations, in seconds, of each event occurrence within the midi file.

### 3.1 Examples

The examples of the 1-7...4-3 pattern shown in Figures 1 and 2 were the result of the following parameter specification:

1. Number of events in the sequence: 4
2. First event: Must include pitches C, E, G, with C on top
3. Second event: Must include pitches G, B, D with B on top
4. Third event: Must include pitches G, B, D, F with F on top
5. Fourth event: Must include pitches C, E, G with E on top
6. Maximum duration of the entire sequence: 5 seconds.
7. Maximum distance between the first two events: 0.5 seconds.

**Fig. 1.** Mozart, Violin Concerto No. 6 in Eb, K268, Allegro moderato, measures 24-29

8. Maximum distance between the last two events: 0.5 seconds.
9. Window length: 1 second
10. Onset interval: 0.5 seconds

Figure 1 shows measures 24-29 in the Allegro moderato of Mozart's Violin Concerto No. 6 in Eb, K268. The annotation highlights the melodic line Do=>Si ... Fa=>Mi. The Do and Mi are played over tonic chords, while the Si and Fa are played over dominant-7 chords. The annotation is a possible user interpretation to the information provided by the query. As noted above, the query retrieves pointers to the locations, in seconds, of each schema event within the midi file.

Figure 2 shows measures 1-5 in the Adagio of Mozart's Violin Concerto No. 1 in Bb, K207. The annotation highlights the melodic line Do=>Si ... Fa=>Mi. In this complex, polyphonic case, the melodic line is not always strictly aligned with the chord progression  $I \Rightarrow V_7 \Rightarrow V_7 \Rightarrow I$ . A fairly wide window length of 1 second allowed for the retrieval of this example. The first Do in measure 2 plays over a tonic chord, which, on the downbeat of measure 3 shifts to a dominant-7, however the Do sustains throughout the first beat, with Si arriving on the second beat. This creates a special tension, which is satisfied by the end of measure 3. In measure 4, the strings play the Fa against a dominant-7. The Mi arrives with the tonic chord on the downbeat of measure 5.

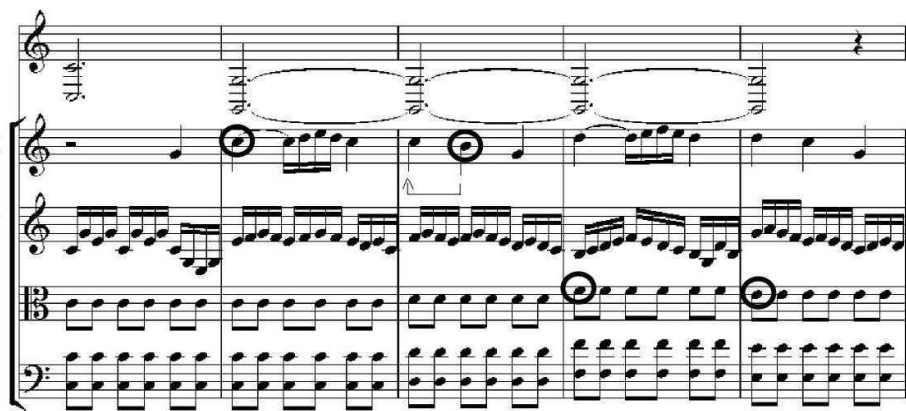


Fig. 2. Mozart, Violin Concerto No. 1 in Bb, K207, Adagio, measures 1-5

## 4 Conclusions and Future Work

The musical data representation model described in this paper has demonstrated its ability to support varied information tasks, using SQL-based retrieval and data mining-based rule extraction. The ability to retrieve structural musical information from midi files creates possibilities for utilizing the large collections of midi files available today towards music research. Time intervals are the basic building blocks of the representation, so a potential for future application to audio data exists and will be explored.

This model could serve as a building block in the task of creating a large-scale corpus of music materials that is accessible to researchers in the Music Information Retrieval (MIR) and Music Digital Library (MDL) communities. For a comprehensive discussion of this goal, the reader is referred to [15].

Future investigation would also include the examination of other musical features such as rhythm, meter and contour in the time series representation.

## Acknowledgements

This work is supported by National Science Foundation (NSF) grant IIS-0328471.

## References

1. Rolland P. Y., Ganascia J. G. "Musical pattern extraction and similarity assessment" in Readings in music and artificial intelligence ed. E. R. Miranda., 2000
2. Huron, D. "Music Research Using Humdrum", Center for Computer Assisted Research in the Humanities, Ohio State University, Columbus, Ohio, 1999
3. Meek C., Birmingham W. P. "Thematic Extractor", ISMIR 2001, Bloomington, Indiana, 2001
4. Downie J. S., Nelson M. "Evaluation of a Simple and Effective Music Information Retrieval Method" in 23<sup>rd</sup> Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Athens, Greece, 2000
5. Doraisamy S., Ruger S. "Robust Polyphonic Music Retrieval with N-grams" in Journal of Intelligent Information Systems, 21:1, 53-70, 2003
6. Birmingham W. P., Dannenberg R. B., Wakefield G. H. Bartsch M., Bykowski D., Mazzoni D., Meek C., Mellody M., Rand W. "Musart: Music Retrieval Via Aural Queries", ISMIR 2001, Bloomington, Indiana, 2001
7. McNab, R. J., Smith L.A., Witten I. H., Henderson C. L., Cunningham S. J. "Towards the Digital Music Library: Tune Retrieval from Acoustic Input" in Digital Libraries, 1996
8. Rumelhart D. E. "Schemata: The Building Blocks of Cognition" in R. J. Spiro, B. C. Bruce, W. F. Brewer (eds.) Theoretical Issues in Reading Comprehension, Erlbaum, Hillsdale, New Jersey, 1980
9. Meyer L. B. Explaining Music: Essays and Explorations The University of Chicago Press, 1973, p. 213
10. Gjerdingen R. O. A Classic Turn of Phrase: Music and the Psychology of Convention, University of Pennsylvania Press, Philadelphia, 1988
11. Gjerdingen R. O. "Courtly Behavior" in Music Perception 13, 1996
12. Classical Music Archives, <http://www.classicalarchives.com>
13. Agrawal R., Srikant R. "Mining Sequential Patterns" in Proceedings of the 11th International Conference on Data Engineering, Taipei, Taiwan, 1995
14. SAS version 9.1
15. Downie, J. S., Futrelle, J., Tcheng D. "The International Music Information Retrieval Systems Evaluation Laboratory: Governance, Access, and Security", ISMIR 2004, Barcelona, Spain, 2004

# A Taxonomy and Representation of Sources of Uncertainty in Active Systems

Segev Wasserkrug<sup>1,2</sup>, Avigdor Gal<sup>2</sup>, and Opher Etzion<sup>1</sup>

<sup>1</sup> IBM Haifa Research Lab, Mount Carmel, Haifa 31905, Israel  
`{segev, Opher}@il.ibm.com`

<sup>2</sup> Technion - Israel Institute of Technology, Haifa 32000, Israel  
`segev@tx.technion.ac.il, avigal@ie.technion.ac.il`

**Abstract.** In recent years, there has been an increased need for the use of active systems - systems that include substantial processing which should be triggered by events. In many cases, however, there is an information gap between the actual occurrences of events to which such a system must respond, and the data generated by monitoring tools regarding these events. For example, some events, by their very nature, may not be signaled by any monitoring tools, or the inaccuracy of monitoring tools may incorrectly reflect the information associated with events. The result is that in many cases, there is uncertainty in the active system associated with event occurrence. In this paper, we provide a taxonomy of the sources of this uncertainty. Furthermore, we provide a formal way to represent this uncertainty, which is the first step towards addressing the aforementioned information gap.

## 1 Introduction and Motivation

In recent years, there has been a growing need for the use of active systems – that is, systems required to act automatically based on events. Examples of events include updates to a database, a room temperature reaching a certain threshold, and a bank account being overdrawn by more than a certain amount. The earliest active systems were active databases, *i.e.* databases in which automatic actions were required as a result of DB operations such as insertions, updates, and deletions. Currently, a major need for such active functionality is for systems in the realms of *Business Activity Monitoring (BAM)* and *Business Process Management (BPM)*. These are systems that monitor, streamline, and optimize the processes and activities which are at the heart of every business enterprise, and they comprise one of the largest of today's emerging markets.

In order for an active system, in any domain, to react automatically to all events of interest, it must first be able to recognize these events. However, in many cases, there is a gap between the actual occurrences of events to which the system must respond, and the data generated by monitoring tools regarding these events. This gap results in **uncertainty**.

*Example 1.* A thermometer generates an event whenever the temperature rises above 37.5°C. The thermometer is known to be accurate to within  $\pm 0.2^\circ\text{C}$ .

Therefore, when the temperature measured by the thermometer is  $37.6^{\circ}\text{C}$ , there is some uncertainty regarding whether the event has actually occurred.

Another gap between the actual occurrence of events and the information available to the system is caused by the following: The information regarding the occurrence of some events (termed *explicit events*) is signalled by *event sources* (e.g., monitoring tools such as the thermometer in Example 1). However, there are other events regarding whose occurrence explicit notification is never sent (events for which explicit notification is sent are termed *explicit events*, while others are termed *non-explicit events*). An example of a non-explicit event is money laundering: Although money laundering either does, or does not, take place, no explicit signal regarding such an event is generated.

For an active system to respond to non-explicit events requires, in many cases, that the occurrences of these events be inferred based on the occurrence of other events (these events are termed *inferred events*). To facilitate such inference, several *event composition languages* have been defined (see Section 2) that make it possible to define a set of complex temporal predicates. However, uncertainty associated with explicit events also propagates to inferred events as well, as the following example shows:

*Example 2.* Consider a rule that states that event  $e_3$  must be inferred whenever an event  $e_2$  occurs after an event of type  $e_1$ . Moreover, assume that it is known that both an event of type  $e_1$  and an event of type  $e_2$  have occurred, but there is uncertainty about the exact time of their occurrence, as follows: The occurrence of  $e_1$  is known to be between time 2 and time 5, and the occurrence of  $e_2$  is known to be between time 3 and time 7. Note that even though the rule is deterministic, the uncertainty regarding the occurrence time of  $e_1$  and  $e_2$ , and the fact that the occurrence of  $e_3$  must be inferred based upon these occurrence times, generates uncertainty with regard to the occurrence of the event  $e_3$ .

There are several possible ways to deal with uncertainty. The first (and currently the most prevalent) way is to ignore it. Ignoring uncertainty greatly simplifies the implementation of active systems. Handling uncertainty in this manner assumes (sometimes implicitly) that in the context in which the active system is used, uncertainty may be ignored without adversely affecting the outcome of the application. As an example, consider once more Example 1. A possible context for such an event is in a hospital ward, where the thermometer is monitoring the body temperature of a patient, and the purpose of the event is to alert medical staff to the possible need for medication. In this context, the uncertainty associated with the actual temperature can usually be safely ignored.

Another way to deal with uncertainty, one that has been explored in the literature (again, see Section 2), is the following: Events are generated only when they occur with absolute certainty. Events regarding whose occurrence there is uncertainty (such as the one in Example 1) are recognized, but are dealt with using a mechanism external to the active system (e.g., a human operator). In this method, the event described in Example 1 would only be generated when the temperature read is above  $37.7^{\circ}\text{C}$ , i.e., in cases in which it is certain that



the temperature is above 37.5°C. In addition, whenever the thermometer reads a temperature between 37.4°C and 37.6°C, the **possible** occurrence of such an event is signaled, and is dealt with outside the application.

A shortcoming of both methods is that uncertainty is not dealt with either explicitly or quantitatively within the active system. Explicit quantification is useful for carrying out automated actions based on mechanisms such as utility theory. Automation becomes mandatory for systems such as security systems which must respond within a very short time span to events such as Denial of Service (DoS) attacks, and security breaches (it is worth noting that both DoS attacks and security breaches are examples of events which are neither signalled nor can be inferred with absolute certainty). BAM and BPM systems are additional examples of systems in which it is an asset to be able to respond automatically to uncertain information regarding events.

Therefore, quantitative and explicit handling of event uncertainty within active systems can greatly enhance these systems' abilities. However, this requires an inference mechanism regarding the occurrence of inferred events and event composition languages that can quantitatively handle uncertain information. A first step towards creating such languages is a formal definition of a quantitative representation of uncertainty. To ensure that such a representation is capable of capturing all the sources of uncertainty, the possible sources of uncertainty must first be listed and possibly classified. Therefore, in this article we provide a taxonomy of possible sources of uncertainties relating to event occurrence. We then describe a data structure that enables capturing this uncertainty quantitatively at the level of each individual event. To the best of our knowledge, ours is the first work that attempts to list and classify the sources of uncertainty associated with events in active systems, and that provides a formal framework for representing these uncertainties.

The rest of this article is organized as follows: In section 2 we present related work. Section 3 provides a taxonomy regarding event uncertainty. Section 4 formally defines a quantitative framework for representing the uncertainty associated with events. We conclude in Section 5.

## 2 Related Work

This section describes two types of relevant work. Works regarding uncertainty in the context of event composition are described in Section 2.1. Section 2.2 discusses quantitative formalisms for representing and reasoning about uncertainty. Such formalisms may serve as a possible basis for representing the event related uncertainty discussed in this work.

### 2.1 Existing Works on Event Composition Uncertainty

Very little work exists regarding uncertainty in the context of event composition. The overwhelming majority of works in this area either completely ignore the possibility of uncertainty in event composition (*e.g.* [7] and [1]), or address it in a very cursory fashion (*e.g.* [2]). The work in [2], for example, enables each event

to have a certainty measure associated with it. However, the semantics of this certainty measure are not formally defined, and as a result, the measure is, in practice, never used. To the best of our knowledge, only the work in [8] begins to provide a more formal treatment of such uncertainty. However, this work does not describe the sources and types of uncertainties associated with events. An additional relevant work is [4]. This work addresses uncertainty resulting from the limited accuracy of clock synchronization in distributed systems. In so doing, it highlights one type of uncertainty, that associated with the time at which the event occurred. In addition, this work proposes that such uncertainty be handled by the following process: Whenever the temporal uncertainty does not propagate to the inferred events, new events may be inferred by the event composition system. If, on the other hand, it is possible that this temporal uncertainty does propagate, a process external to the event composition system (*e.g.*, a human operator) is triggered to resolve this uncertainty. Therefore, even the single type of uncertainty (uncertainty regarding the occurrence time of events) which is addressed in this work is not handled, much less quantitatively represented, within the scope of an event composition system.

## 2.2 Formalisms for Quantitative Representation and Reasoning About Uncertainty

The most well known framework for quantitative representation and reasoning about uncertainty is probability. Other quantitative approaches include, among others, *Lower and Upper Probabilities*, *Dempster-Shafer Belief Functions*, and *Possibility Measures* (see [3]). In addition, *Fuzzy Sets* and *Fuzzy Logic* [9] are often considered for dealing with uncertainty.

In this work, to represent the uncertainty associated with events we have chosen to use probability theory. We made this decision for the following reasons:

- Probability theory is a well-understood and powerful tool.
- Under certain assumptions, probability is the only "rational" way to represent uncertainty (again, see [3]).
- There are well-known and accepted methodologies for carrying out inferences based on probability theory, involving structures known as Bayesian Networks (see [5]).
- Probability theory can be used together with utility theory (see [6]) for automatic decision making. Such automatic decision making would facilitate the implementation of automatic actions by active systems.

## 3 Taxonomy of Event Uncertainty

This section provides a taxonomy of event uncertainty as follows: Section 3.1 lays some groundwork by formally describing an event and its related data. Section 3.2 defines two dimensions that can be used to classify the uncertainties as relating to events, and Section 3.3 describes the causes of event uncertainties for the defined dimensions.

### 3.1 Event Definition

A formal description of the concept of an event in active systems appears below in Definition 1:

**Definition 1.** An *event* is an **actual** occurrence or happening that is **significant** (falls within a domain of interest to the system), **instantaneous** (takes place at a specific point in time), and **atomic** (it either occurs or not) .

Examples of events include notifications of changes in a stock price, failures of IT components, and a person entering or leaving a certain geographical area.

A variety of data can be associated with the occurrence of an event. Two examples of such data are: The point in time at which an event occurred, and the new price of a stock in an event describing a change in the price of a specific stock. Some data are common to all events (e.g. their time of occurrence), while others are specific only to some events (e.g. data describing stock prices are relevant only for stock-related events). The data items associated with an event are termed *attributes*.

Due to the information gap described in Section 1, it is necessary to differentiate between the actual event, and the information the system has regarding the event. In order to enable this differentiation, we define the notion of an *Event Instance Data (EID)*, which is the data structure that contains the information the system has regarding each event.

### 3.2 Dimensions of Event Uncertainty

When describing sources of event uncertainty, it is useful to classify the uncertainty according to two orthogonal dimensions: *Element Uncertainty* and *Origin Uncertainty*. The first dimension, Element Uncertainty, refers to the fact that event-related uncertainty may involve one of two elements:

1. *Uncertainty regarding event occurrence*: Such uncertainty is associated with the fact that although the actual event occurrence is atomic, i.e. the event either did or did not occur (see Definition 1), the active system does not know whether or not this event has in fact occurred. One example of such an event is the thermometer reading event from Example 1. Another example is money laundering, i.e. at any point in time, money laundering either was, or was not, carried out by some customer. However, an active system can probably never be certain whether or not money laundering actually took place.
2. *Uncertainty regarding event attributes*: Even in cases in which the event is known to have occurred, there may be uncertainty associated with its attributes. For example, while it may be known that an event has occurred, its time of occurrence may not be precisely known.

The second dimension, *Origin Uncertainty*, pertains to the fact that in an active system, there may be two types of events: Explicit events, signalled by

event sources, and inferred events, which are inferred based on other events (see Section 1). In the following, events which serve as the basis for the inference of other events will be termed *evidence events*. Therefore, there are two possible origins for uncertainty:

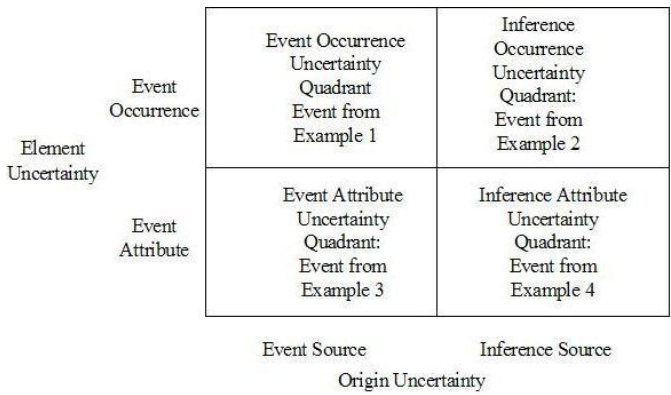
1. *Uncertainty originating at the event source*: When an event originates at an event source, there may be uncertainty associated either with the event occurrence itself, or the event's attributes, due to a feature of the event source. Example 1, in which uncertainty regarding an event occurrence is caused by the limited measuring accuracy of a thermometer, illustrates such a case.
2. *Uncertainty resulting from event inference*: Due to some events being inferred based on other events, uncertainty can propagate to the inferred events. This is demonstrated by Example 2, in which there is uncertainty regarding the occurrence of event  $e_3$  due to the uncertainty regarding the occurrence time of events  $e_1$  and  $e_2$ .

Example 1 demonstrates an event whose uncertainty originates at the event source, and concerns the actual occurrence of an event. Similarly, Example 2 describes an event whose uncertainty results from inference, and, again, involves the actual occurrence of an event. Below, two additional examples are provided: Examples 3 and 4. In Example 3, the uncertainty of an event originates from the source, but is limited to its attributes, rather than to its occurrence. Example 4 shows uncertainty regarding an event's attributes resulting from event inference.

*Example 3.* Consider a case in which an event must be generated whenever the temperature read by a thermometer changes. Assume that the thermometer under discussion has the same accuracy as the one defined in Example 1. Furthermore, assume that the new temperature is an attribute of this event. In this case, there is no uncertainty regarding the actual occurrence. There is only uncertainty regarding this new temperature attribute.

*Example 4.* Consider a case similar to the one in Example 2, in which  $e_3$  should be inferred whenever an event  $e_2$  occurs after an event of type  $e_1$ . Assume that the inferred event  $e_3$  has an attribute  $a_1^3$  whose value is the sum of the value of the attribute  $a_1^1$  of event  $e_1$  and the value of the attribute  $a_1^2$  of event  $e_2$ . Now assume that both  $e_1$  and  $e_2$  are known to have occurred with certainty, but there is uncertainty regarding the value of attribute  $a_1^1$ . In this case there is uncertainty only regarding the value of attribute  $a_1^3$ , and this uncertainty results from event inference.

The above examples demonstrate that the two dimensions are indeed orthogonal. Therefore, uncertainty associated with events could be mapped into one of four quadrants, as shown in Figure 1. In addition, due to the orthogonality of these dimensions, we define four types of event uncertainty: *Uncertainty regarding event occurrence originating at an event source*, *uncertainty regarding event occurrence resulting from inference*, *uncertainty regarding event attributes originating at an event source*, and *uncertainty regarding event attributes resulting from event inference*.



**Fig. 1.** Display of quadrants defined by uncertainty dimensions, as well as the mapping of the events in Examples 1-4 to these quadrants

3.3 Causes of Event Uncertainty

This section describes, at a high level, the various causes of event uncertainty, according to the dimensions defined in Section 3.2.

**Causes of Uncertainty Originating at the Source.** Uncertainty regarding event *occurrence* originating at an event source is caused by one of the following:

1. **An unreliable source:** An event source may malfunction and indicate that an event has occurred even if it has not. Similarly, the event source may fail to signal the occurrence of an event which has, in fact, occurred.
2. **An imprecise event source:** An event source may operate correctly, but still fail to signal the occurrence of events due to limited precision (or may signal events that did not occur). This is illustrated by Example 1.
3. **Problematic communication medium between the event source and the active system:** Even if the event source has full precision, and operates correctly 100% of the time, the communication medium between the source and the active system may drop indications of an event’s occurrence, or generate indications of events that did not occur.
4. **Uncertainty due to estimates:** In some cases, the event itself may be a result of a statistical estimate. For example, it may be beneficial to generate an event whenever a network Denial of Service (DoS) event occurs, where the occurrence of such a DoS event is generated based on some mathematical model. However, as the mathematical model may produce erroneous results, this event also has uncertainty associated with it.

Uncertainty regarding the *attributes* originating at the event source can also be caused by any of the above reasons. An unreliable or imprecise source may be unreliable or imprecise regarding just the attribute values. Similarly, the

communication medium may garble just the values of attributes, rather than the fact that the event did or did not occur. Finally, estimates may also result in uncertainty regarding event attributes.

In distributed systems, there exists an additional cause for uncertainty regarding the special attribute capturing the occurrence time of the event. This is due to the fact that in distributed systems, the clocks of various nodes are usually only guaranteed to be synchronized to within some interval of a global system clock (see [4]). Therefore, there is uncertainty regarding the occurrence time of events as measured according to this global system clock.

Note that in both of the above cases, uncertainty regarding a specific event may be caused by a combination of factors. For example, it is possible that both the event source itself and the communication medium simultaneously corrupt information sent regarding the same event.

**Causes of Inferred Uncertainty.** Uncertainty regarding event *occurrence* resulting from inference has the two possible causes:

1. **Propagation of Uncertainty:** An inferred event can be a result of a deterministic rule. However, when there is uncertainty regarding the events based on which the new event is inferred, there is uncertainty regarding the inferred event. This is depicted in Example 2.
2. **Uncertain Rules:** The rule itself may be defined in an uncertain manner, whenever an event cannot be inferred with absolute certainty based on other events. An example of this is money laundering, where events denoting suspicious transactions only serve to indicate the possible occurrence of a money laundering event. Usually, a money laundering event cannot be inferred with certainty based on such suspicious transactions.

Note that these two causes may be combined. That is, it may happen that not only is the inference of an event based on an uncertain rule, but also uncertainty exists regarding the occurrence (or attribute values) of one of the events which serve as evidence for this inference.

Regarding uncertainty of inferred event *attributes*, the possible causes depend on how these attributes are calculated from the attributes of the evidence events. The most intuitive way to calculate such inferred attributes is using deterministic functions defined over the attributes of the evidence events. In such a case, the only cause of uncertainty in the inferred attributes is propagation of uncertainty from the attributes of the evidence events. This is because the uncertainty regarding the event attributes is defined as the uncertainty regarding the attribute values given that the event occurred. Therefore, the rule cannot induce uncertainty regarding the attribute values of the inferred events. However, if the inference system makes it possible to define the attribute values of the inferred events in a non-deterministic manner, uncertain rules may also be a cause of inferred attribute uncertainty.

Figure 2 summarizes the causes of uncertainty

|  |   |   |
|--|---|---|
| Uncertainty<br>Regarding Event<br>Occurrence | <ul style="list-style-type: none"> <li>• Unreliable Source.</li> <li>• Imprecise Source.</li> <li>• Problematic Communication Medium</li> <li>• Estimates</li> </ul>  | <ul style="list-style-type: none"> <li>• Propagation of Uncertainty</li> <li>• Uncertain Rules</li> </ul> |
| Uncertainty<br>Regarding Event<br>Attributes | <ul style="list-style-type: none"> <li>• Unreliable Source.</li> <li>• Imprecise Source.</li> <li>• Problematic Communication Medium</li> <li>• Estimates</li> <li>• Time Synchronization in Distributed Systems</li> </ul> | <ul style="list-style-type: none"> <li>• Propagation of Uncertainty</li> </ul>                            |
|  | Origin: Event Source  | Origin: Event Inference   |

Fig. 2. Causes of uncertainty

## 4 Representing Uncertainty

As stated in Section 2.2, probability theory has been chosen in this work to as an uncertainty representation mechanism. Section 4.1 how probability theory can be used to represent the uncertainty associated with events. In addition, any quantitative approach based on probability theory requires the quantification of various probabilities. Section 4.2 discusses how using the taxonomy defined in this work can help in quantifying these probabilities.

### 4.1 Representing Uncertainty of Individual Events

As defined in Section 3.1, for each event, the information held by the event composition system is contained in a data structure called an EID. This EID must be capable of incorporating all relevant data about an event, including its type, attributes such as times of occurrence, and uncertainty quantification. In active systems with no uncertainty, each event can be represented by a single tuple of values  $Val = \langle val_1, \dots, val_n \rangle$  (one value for each type of data associated with the event).

To capture the uncertainty associated with an event instance, we extend each EID to be a Random Variable ( $RV$ ) such that the possible values of EID are taken from the domain  $V = \{notOccurred\} \cup V'$ , where  $V'$  is a set of tuples of the form  $\langle val_1, \dots, val_n \rangle$ . In addition, the semantics associated with the probabilities on this RV are the following: The probability that the value of  $E$  belongs to a subset  $S \subseteq V \setminus \{notOccurred\}$  is the probability that event  $e$  has occurred, and that the value of its attributes is some tuple of the form  $\langle val_1, \dots, val_n \rangle$ , where  $\{\langle val_1, \dots, val_n \rangle\} \in S$ . Similarly, the probability associated with the value  $\{notOccurred\}$  is the probability that the event did not occur. Such a RV is illustrated in Example 5.

*Example 5.* In case of the thermometer related event appearing in Example 1, assume the following: whenever the thermometer reads  $37.5^\circ\text{C}$ , the probability that the temperature is  $37.3^\circ\text{C}$  is 0.1, the probability that the temperature is

37.4°C is 0.2, the probability that the temperature is 37.5°C is 0.5, the probability that the temperature is 37.6°C is 0.2, and the probability that the temperature is 37.7°C is 0.1. Therefore, the probability that the event did not occur is 0.3 and the probability that the event did occur is 0.7. Moreover, assume that whenever the event does occur, the temperature is an attribute of this event.

Assume that at time 5, the thermometer registers a reading of 37.5°C. This information would be represented by an EID  $E$  with the following set of values:  $\{notOccurred\}$  - indicating that the event did not occur,  $\{5, 37.5^\circ C\}$  - indicating that the event occurred at time 5 with temperature 37.5°C,  $\{5, 37.6^\circ C\}$  - indicating that the event occurred at time 5 and the temperature was 37.6°C, and  $\{5, 37.7^\circ C\}$ , indicating that the event occurred at time 5 with temperature 37.7°C. Moreover, the probabilities defined over  $E$  are the following:  $\Pr(E = \{notOccurred\}) = 0.3$ ,  $\Pr(E = \{5, 37.5^\circ C\}) = 0.5$ ,  $\Pr(E = \{5, 37.6^\circ C\}) = 0.2$  and  $\Pr(E = \{5, 37.7^\circ C\}) = 0.1$ .

Note that, by definition, the set of possible values of the EID RVs contain information regarding both the occurrence of the event and its attributes. Therefore, this representation is sufficient to represent the uncertainty regarding both occurrence and attributes.

## 4.2 Inferring the Probabilities

The probabilities representing the various uncertainties must be quantified for the above formal representation of event uncertainty. For uncertainties resulting from event inference, these probabilities depend on the definition and semantics of the rules. Therefore, their quantification will not be discussed in this work (for an example of such rules and semantics, see [8]).

For uncertainty originating at the event source, the uncertainty type (i.e. whether the actual occurrence of the event or an attribute value of the event) defines the probabilities that must be quantified, as follows:

1. **Uncertainty regarding event occurrence originating at the event source:** In this case, there is uncertainty because one of the following may occur: Either the actual event occurs but is not signaled, or a signal regarding the occurrence of an event is generated without the actual event having occurred. Therefore, once it is recognized that this is a cause of uncertainty, it is clear that the following probabilities must be quantified: The probability that the event occurred but was not signalled, and the probability that the event was signalled without such an event occurring. More detailed information regarding the exact source of uncertainty can be used to assist in the quantification of these probabilities. For example, if the cause is an unreliable event source, reliability information provided by the manufacturer of the event source may be used to quantify the probability. Likewise, if the source is a problematic communication medium, information regarding the network topology and the components in the network may provide useful information for this quantification.



- 2. Uncertainty regarding event attributes originating at an event source:** The uncertainty in this case arises from either of the following: Either an incorrect value of the attribute is provided, or no value for the attribute is provided at all. In the first case (i.e., incorrect value), a probability space that quantifies the probability of all possible values of the attribute given the value signalled by the event source must be provided. In the second case, a probability distribution of all possible values of the attribute is required. Again, more detailed information regarding the exact source of uncertainty may help both in defining the set of all possible values of the attributes, and providing the required probability space. To illustrate this, consider Example 5: Knowledge of the imprecision inherent in the thermometer makes it possible to define the value of possible ranges (e.g.,  $37.3^{\circ}\text{C} - 37.7^{\circ}\text{C}$  when the thermometer registered a temperature of  $37.5^{\circ}\text{C}$ ). Similarly, in the case of time in distributed systems, the reading registered by a local clock on a system participating in a time synchronization algorithm is known to be within some interval of a global time (see [4]).

It is worth noting that in all of the above cases, available knowledge regarding the event source may be insufficient to uniquely define the required probabilities. In such cases, techniques such as the *Principle of Indifference* (see [3]) may be used for probabilistic quantification (the Principle of Indifference assumes that all outcomes are equally likely given no evidence to the contrary).

In many cases, the event source itself is not capable of producing the representation of an EID as defined in Section 4.1. Therefore, a component that transforms the information received by an event source to the EID format required by the active system would be useful. This is depicted in Figure 3. In this architecture, the *Event to EID Translator* component has the responsibility of transforming the information generated by the event source to the Random Variable EID format defined in Section 4.1.



**Fig. 3.** Probability Augmentation Architecture

To understand how this component would work, consider again the thermometer event defined in Example 1. In this case, the event source would generate an event whenever the temperature of the thermometer exceeds  $37.5^{\circ}\text{C}$ , with an attribute detailing the temperature registered. If, for instance, the registered temperature is  $37.5^{\circ}\text{C}$ , the Event to EID translator would create an EID as defined in Example 5.

The Event to EID translator would need to have knowledge of the specific event source for each event for the EID to be correctly created. In addition, assumptions such as the Principle of Indifference may also have to be incorporated.

## 5 Summary, Discussion and Future Work

In this paper, a taxonomy of the uncertainty relating to events in active systems was presented according to two orthogonal dimensions: Element Uncertainty, i.e., whether the uncertainty involves the actual occurrence of an event or an attribute of it; and Origin Uncertainty, which specifies whether the uncertainty originates from an event source or is a result of event inference. The value of this taxonomy is that it clearly maps the possible types of uncertainties associated with events in active systems. Such a formal definition is a prerequisite to ensuring that a quantitative representation sufficiently captures all the required information about such uncertainty. In addition, a formal representation based on probability theory was defined, and it was shown how the uncertainty taxonomy is useful in quantifying the required probability.

There are many avenues open for future research. For example, while the taxonomy provides some assistance in the quantification of probabilities, additional work is required to ensure that such quantification can always be obtained or estimated. In addition, future research should also be carried out to consider alternative representations of uncertainty, such as Dempster-Shafer belief functions. Such alternate representations may be more appropriate in some settings, and may be better suited to deal with a lack of quantitative information.

## References

1. A. Adi, D. Botzer, and O. Etzion. The situation manager component of amit - active middleware technology. In *NGITS '02: Proceedings of the 5th International Workshop on Next Generation Information Technologies and Systems*, pages 158–168. Springer-Verlag, 2002.
2. A. Adi and O. Etzion. AMIT - the situation manager. *VLDB Journal*, 13(2):177–203, 2004.
3. J. Y. Halpern. *Reasoning About Uncertainty*. MIT Press, 2003.
4. C. Liebig, M. Cilia, and A. Buchman. Event composition in time-dependant distributed systems. In *Proceedings of the Fourth IFCIS International Conference on Cooperative Information Systems*, pages 70–78. IEEE Computer Society Press, 1999.
5. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman, 1988.
6. H. Raiffa. *Decision Analysis : Introductory Lectures on Choices under Uncertainty*. Addison-Wesley, 1968.
7. C. S. Snoop: An expressive event specification language for active databases. *Data and Knowledge Engineering*, 14(1):1–26, 1994.
8. S. Wasserkrug, A. Gal, and O. Etzion. A model for reasoning with uncertain rules in event composition systems. In *Proceedings of the 21st Annual Conference on Uncertainty in Artificial Intelligence (UAI-05)*, pages 599–606. AUAI Press, 2005.
9. L. A. Zadeh. Fuzzy sets. *Information Control*, 8:338–353, 1965.

# Analyzing Object-Oriented Design Patterns from an Object-Process Viewpoint

Galia Shlezinger<sup>1</sup>, Iris Reinhartz-Berger<sup>2</sup>, and Dov Dori<sup>1</sup>

<sup>1</sup> Faculty of Industrial Engineering and Management,  
Technion-Israel Institute of Technology, Haifa 32000, Israel  
galias@tx.technion.ac.il, dori@ie.technion.ac.il

<sup>2</sup> Department of Management Information Systems,  
University of Haifa, Haifa 31905, Israel  
iris@mis.haifa.ac.il

**Abstract.** Design patterns are reusable proven solutions to frequently occurring design problems. To encourage software engineers to use design patterns effectively and correctly throughout the development process, design patterns should be classified and represented formally. In this paper, we apply Object Process Methodology (OPM) for representing and classifying design patterns. OPM enables concurrent representation of the structural and behavioral aspects of design patterns in a single and coherent view. Comparing OPM and UML models of seven popular design patterns, we found that the OPM models are more compact, comprehensible and expressive than their UML counterparts. Furthermore, the OPM models induce a straightforward classification of these design patterns into four groups: creational, structural composition, wrapper, and interaction design patterns.

## 1 Introduction

Design Patterns describe generic solutions for recurring problems to be customized for a particular context. They have attracted the interest of researchers and practitioners as proven reusable solutions to frequently occurring design problems. Shalloway and Trott [19] suggested several reasons for using, studying, and dealing with design patterns, including the reuse of existing, high-quality solutions and establishing common terminology to improve communication within teams. However, deploying these solutions to develop complex information systems is a tedious task that involves integration issues and iterative development. It is, hence, important to describe design patterns, the problems they intend to solve, the context in which they can be reused, and their consequences in a formal, unambiguous way that can be easily understood by designers. The idea of identifying and reusing the common characteristics of a problem has also been adapted in other fields; one such example is generic tasks [4] in knowledge-based systems.

The increasing number of design patterns that have been proposed and published over the last decade emphasizes the need for a good design patterns representation language, as well as a framework for organizing, classifying, categorizing, and evaluating design patterns, which will help designers in choosing and implementing the

correct solutions to the problems at hand. In this work, we suggest Object Process Methodology (OPM) [6] for both purposes. OPM, which supports two types of equally important elements, objects and processes, enables the representation of both structural and behavioral aspects of design patterns in a single coherent view. Furthermore, the OPM design pattern models lend themselves naturally to a clear, useful classification. This classification is directly derived from the OPM models and does not require justifications and further explanations in plain text. The contribution of our work is therefore two-fold: First, we apply OPM to model and portray the essence of design patterns in a more direct, complete, and comprehensible way than what can be done using object-oriented languages. The completeness of the pattern models is due to OPM's ability to represent both the structure and the behavior of the patterns. Secondly, the categories of design patterns defined in this work are solidly grounded by distinctive characteristics of their respective OPM models. Based on this classification, design patterns can be used regardless of the chosen modeling language.

The rest of the paper is structured as follows. In Section 2 we review work relevant to design pattern languages and classification frameworks. Section 3 provides a short overview of OPM, while Section 4 presents OPM models of several design patterns, making the point that these models induce an accurate design pattern classification scheme. Section 5 concludes our work.

## 2 Motivation and Background

As noted, design patterns provide solutions for recurring design problems. The idea of documenting design solutions as patterns is attributed to the American architect Christopher Alexander [1]. Applying his idea to object-oriented design and programming, design patterns are usually described using a template. The template used by Gamma et al. in [11], for example, consists of *pattern name and classification*, *intent (motivation)*, *applicability*, *structure*, *participants*, *collaboration*, *consequences*, *implementation*, *sample code*, *known uses*, and *related patterns*. The structure of the design pattern is often illustrated by a graphical representation, such as such as OMT [18] or UML [14] (in this work we will not differentiate between OMT and UML class diagrams). Sometimes, the representations are also accompanied by brief textual descriptions of the basic elements (i.e., classes) that compose the design pattern. The behavioral aspect of the design pattern usually gets much less attention, and is sometimes described informally in text or through partial diagrams that specify the collaboration between the various elements. Such semi-formal representations of design patterns hinder their rigorous, systematic use.

Different languages have been proposed to formally represent design patterns. Some employ mathematical descriptions [8], which require a fair amount of mathematical skills and are not easily understood by designers, while others suggest visual representations or markup languages. Several languages for describing design patterns are based on UML or its extensions, e.g., [9, 12]. Their main shortcoming is the lack of formality. Generally speaking, consistency and integrity are Achilles' heel of UML [15, 17]. Markup languages, such as XML and OWL [5, 16], are also used for describing design patterns. While this approach may be very useful for building tools

that support design patterns, markup languages are machine-understandable at the expense of being cryptic to humans [7].

Since the number of design patterns is continuously increasing, there is a growing need not only to formally describe the different aspects of design patterns, but also to organize and classify them according to their purpose or structure. Noble [13] and Zimmer [20], for example, have classified relationships among design patterns that are mostly hierarchical. Zimmer's categorization brought him to the conclusion that the Factory Method design pattern is not really a pattern, but rather a manifestation of a "*X uses Y*" relationship between two other design patterns: Abstract Factory and Template Method. He also suggested modeling behaviors as objects and introduced a new design pattern, called Objectifying Behavior, for this purpose.

Gamma et al. [10, 11] have used two dimensions for classifying design patterns: *scope*, which specifies whether the pattern applies to classes or objects, and *purpose*, which reflects the intension of the patterns. According to the purpose dimension, a design pattern can be creational, structural, or behavioral.












Another design pattern classification scheme [2] organizes patterns according to their granularity, functionality, and structural principles. According to this categorization, design patterns are only one group of patterns that belong to a specific level of granularity. Design patterns were further divided into *structural decomposition*, *organization of work*, *access control*, *management*, and *communication* [3].

Since the classification schemes of design patterns are basically object-oriented, their categorization, justified primarily with objects in mind, may not be comprehensive. Adopting Object Process Methodology (OPM), which departs significantly from the object-oriented paradigm, our approach to modeling and categorizing design patterns is fundamentally different. These differences are most evident in design patterns that involve dynamic aspects, since in OPM structure and behavior are equally important. Analyzing the object-oriented classification of the design patterns in [11] with respect to their OPM design pattern models, we offer an improved classification scheme.

### 3 Object-Process Methodology

Object Process Methodology (OPM) is an integrated modeling paradigm to the development of systems in general and information systems in particular. The two equally important class types in OPM, objects and processes, differ in the values of their perseverance attribute: the perseverance value of object classes is static, while the perseverance value of process classes is dynamic. OPM's combination of objects and processes in the same single diagram type is intended to clarify the two most important aspects that any system features: structure and behavior. OPM supports the specification of these features by structural and procedural links that connect things. Structural links express static relations such as aggregation-participation and generalization-specialization between pairs of things. Procedural links, on the other hand, connect things to describe the behavior of a system, i.e., how processes transform, use, and are triggered by objects. More about OPM can be found in [6]. Table 1 summarizes the OPM elements used in this paper.

**Table 1.** Main OPM elements, their symbols, and semantics

| Element Name                  | Symbol  | Semantics   |
|-------------------------------|---|---|
| Object                        |  | A thing that has the potential of unconditional existence   |
| Process                       |  | A pattern of transformation that objects undergo  |
| Instrument link               |  | A procedural link indicating that a process requires an unaffected object (input) for its execution |
| Effect link                   |  | A procedural link indicating that a process changes an object                                       |
| Result link                   |  | A procedural link indicating that a process creates an object                                       |
| Event link                    |  | A procedural link indicating that a process is activated by an event (initiated by an object)       |
| Invocation link               |  | A procedural link indicating that a process invokes another process                                 |
| Structural Link               |  | A structural relation between objects   |
| Aggregation-Participation     |  | A structural relation which denotes that a thing (object or process) consists of other things       |
| Exhibition-Characterization   |  | A structural relation representing that a thing (object or process) exhibits another thing          |
| Generalization-Specialization |  | A structural relation representing that a thing is a sub-class of another thing                     |

## 4 Classification of Design Patterns

In this section we examine the classification of design patterns presented in [11] in terms of OPM. Some of the patterns that are discussed there are modeled using UML class and sequence diagrams. Due to lack of space, we present here only the UML class diagrams of the design patterns.

### 4.1 Creational Design Patterns

Creational design patterns relate to class instantiation. Figures 1 and 2 describe two creational design patterns: *Factory Method* and *Builder*. Each description includes a problem definition, suggested solution, and UML and OPM models.

As the models in these two figures demonstrate, the UML and OPM models of the design patterns differ in both orientation and abstraction level. While the UML models are object-oriented, i.e., they comprise object classes only, the OPM models are composed of both object and process classes. Thanks to the notion of stand-alone processes, the OPM design pattern models do not require supplementary object classes, which are used only as owners of methods (or operations) or as containers of other classes.

Furthermore, OPM enables specification of behaviors. The UML model of the Factory Method design pattern, for example, requires specification of calling the "Factory Method" from "An Operation" and indicating by informal notes that the "Factory Method" returns a "Concrete Product". The OPM model, on the other hand, specifies these requirements formally by using multiplicity constraints (zero to many, as discussed next) on the number of operations before and after the "Factory Method", and a result link, indicating that the "Factory Method" generates and returns a "Product". Multiplicity constraints in OPM can be associated not only to relations but also to object and process classes. A multiplicity constraint on a thing (object or process) in a design pattern model indicates how many occurrences of this thing can appear in an application that implements the design pattern. In the case of the Factory method design pattern, the process "Operation" can appear zero or more times, as indicated by the "0..n" label at the upper left corner of the process, implying that "Factory Method" is called from somewhere within the process called "An Operation", including its very first or last operation.

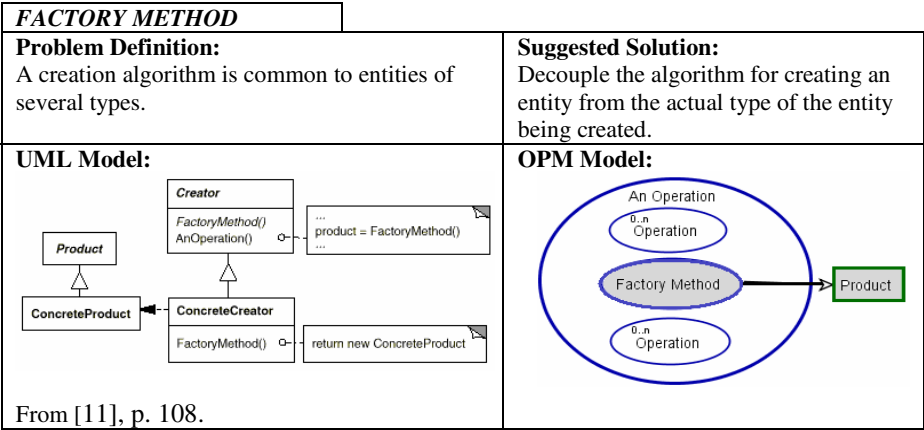
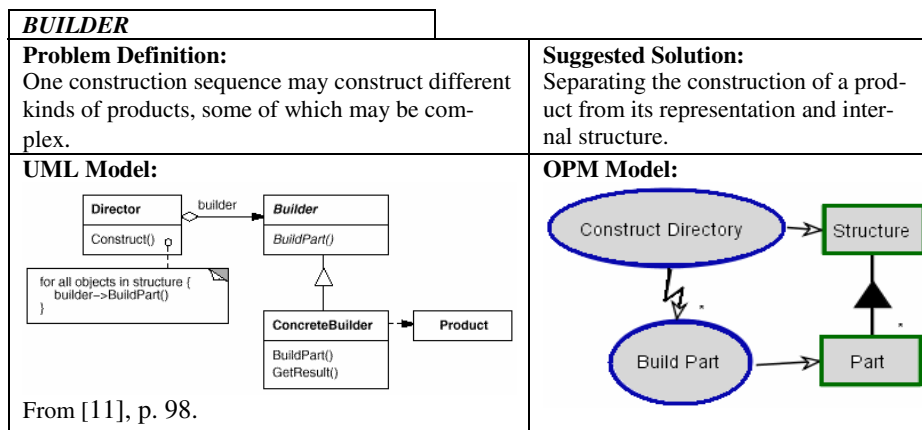


Fig. 1. The Factory Method design pattern

These models also demonstrate the scaling mechanisms that are built into OPM for enabling abstraction and refinement of things. These mechanisms enable detailing an OPM model without losing the "big picture" of the system or pattern being modeled. The mechanism used in this case is in-zooming, in which an entity is shown enclosing its constituent elements. The vertical axis is the time line, so within an in-zoomed process it defines the execution order of the subprocesses, such that subprocesses that need to be executed in a sequence are depicted stacked on top of each other, with the earlier process on top of a later one. "An Operation" is in-zoomed here to display its subprocesses: first a set of zero or more "Operations" is performed, then the "Factory Method" is activated (creating "Products"), and finally another set of 0 or more "Operations" is performed.



**Fig. 2.** The Builder design pattern

The second difference between the UML and OPM design pattern models is in their support of abstraction levels. While in the UML models both the abstract and concrete classes appear in the models, in the OPM models, which are actually meta-models, only the "abstract" classes appear, while the concrete classes that implement the abstract ones appear only in the actual models that make use of the design pattern. In these concrete application models, the application elements will be classified according to the design pattern elements using a mechanism similar to UML stereotyping. This separation of the design pattern model—the OPM metamodel—from the application model results in a more abstract, formal, compact, and comprehensible design pattern model. The OPM design pattern models can therefore be considered as templates for applications that make use of these design patterns or as guiding meta-models taken from some meta-library.

Studying the OPM models of the creational design patterns reported in [11] clearly shows that the basic idea behind creational design patterns is separating the construction logic from the objects. The construction logic can be specified as a process that creates the required object(s) as denoted by a result link from the process to the object. This recurring **process – result link – object** pattern is distinguishable in the OPM models in figures 1 and 2 by the pertinent object and process being marked in grey and with thick lines. This pattern indeed justifies the classification of these two design patterns as creational.

## 4.2 Structural Design Patterns

Structural design patterns relate to class and object composition. They use inheritance to compose interfaces and define ways to compose objects to obtain new functionality. Figures 3 and 4 respectively describe the *Decorator* and *Composite* structural design patterns.

The design patterns in this group further emphasize and clarify the fundamental differences between the UML and OPM design pattern models: All the constraints which are specified in the UML models as notes (in plain text) are expressed formally



in the OPM models. The Decorator design pattern in figure 3, for example, requires adding behavior to a "component". This addition is specified in the UML model as a note in which the "Decorator" "Operation" (which includes the "Component" "Operation") is first called and then followed by the "Added Behavior". However, this is only one possibility defined in the design pattern problem section, which reads: "There is a need to add functionality that *precedes or follows* a basic functionality..." The OPM model enables any number of operations *before and after* the basic functionality, which is called in the model "Component Operation".

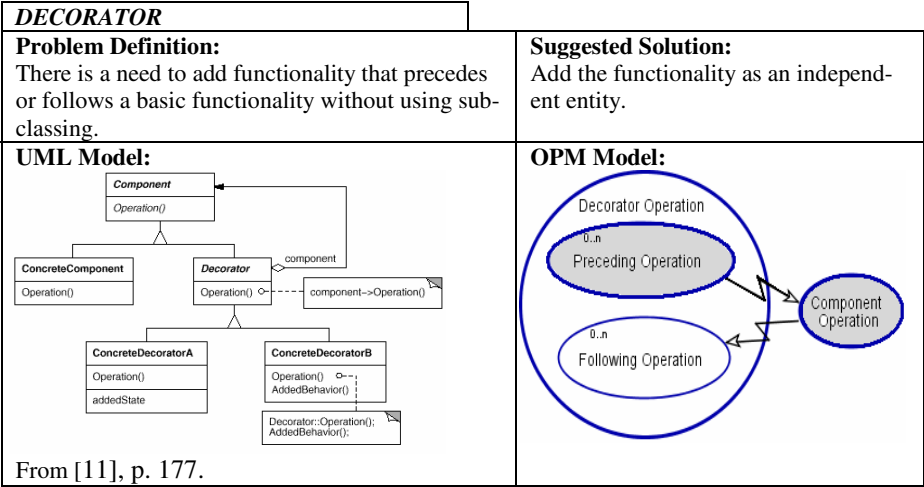


Fig. 3. The Decorator design pattern

The UML and OPM models of the Composite design pattern in figure 4 are completely different. The UML model uses an aggregation relation between "Composite" and "Component", while in the OPM model, "Composite" is zoomed into "Components". However, zooming into an OPM process has both structural aspects (containment, part-whole relations), and procedural aspects (execution order). Furthermore, we have found that the inheritance relation between "Composite" and "Component" in the UML model of this pattern is redundant in the OPM model, since its only meaning is that they are both processes.

The recurrent pattern in the OPM models of this group of design patterns is **process – invocation link – process**. Another observation is that all the design pattern models contain a process which is further zoomed into subprocesses, one of which invokes another process. Although on the surface this pattern should be classified as behavioral rather than structural, a deeper look at the OPM models shows that they basically define the structure of components, each of which is a process. The OPM models of the Factory Method and Decorator design patterns reinforce this observation. The two design patterns are similar in that they both have an operation (performing some function) that may be preceded and/or followed by any number of other

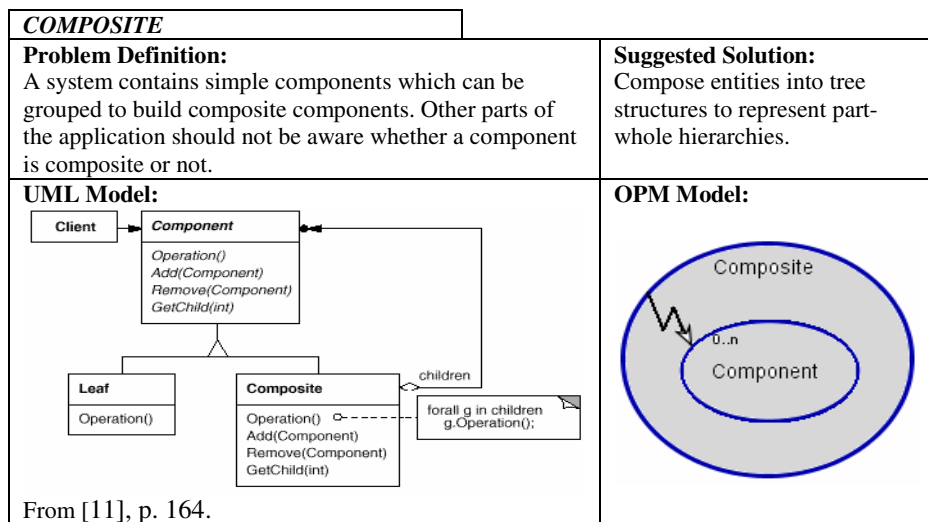


Fig. 4. The Composite design pattern

operations. However, the Factory Method emphasizes the creation of a product, while the Decorator focuses on the separation between the basic functionality—the "Component Operation" and the operation that uses it—the "Decorator Operation".

The OPM model of the Decorator design pattern represents a structural aspect which is common to many behavioral patterns: the invoked process is external to the in-zoomed process. This common aspect justifies their classification by several design pattern classification schemes, including [10], [11], and [19], as *wrappers*. The wrapping essence of the Decorator design pattern is supported by its OPM model, in which one process actually wraps a call to another process.

### 4.3 Behavioral Design Patterns

Behavioral design patterns define algorithms and object responsibilities. They also help in designing communication modes and interconnections between different classes and objects. Figures 5-7 describe three behavioral design patterns: *Chain of Responsibility*, *Observer*, and *Template Method*.

As the OPM models of these behavioral design patterns clearly show, the focus here is on behavior and way of invocation. It should, hence, come as no surprise that most of the OPM models in this category are dominated by processes. In the OPM model of the Chain of Responsibility design pattern, a "Handler" invokes (triggers) its successor. In the OPM model of the Observer design pattern, there are two different processes: "Notify", which is triggered by the "Subject" and affects the relevant "Observers", and "Update", in which the "Observer" can change the state of the "Subject".

The pattern that characterizes most of the behavioral design pattern OPM models is **object – event link – process – effect link – object**, implying that these design patterns have both triggering and affecting aspects. The two exceptions to this

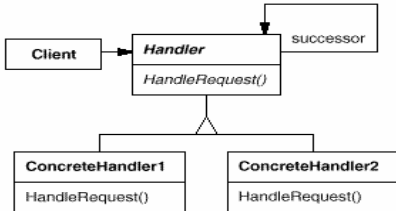
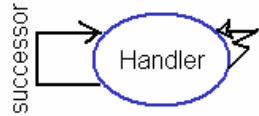
| CHAIN OF RESPONSIBILITY  |   |
|--|---|
| <b>Problem Definition:</b><br>There is a need to invoke a behavior without specifying which implementation should be used. | <b>Suggested Solution:</b><br>Create a chain of behaviors and pass our request along this chain until it is handled |
| <b>UML Model:</b><br>                     | <b>OPM Model:</b><br>              |
| From [11], p. 225.   |   |

Fig. 5. The Chain of Responsibility design pattern

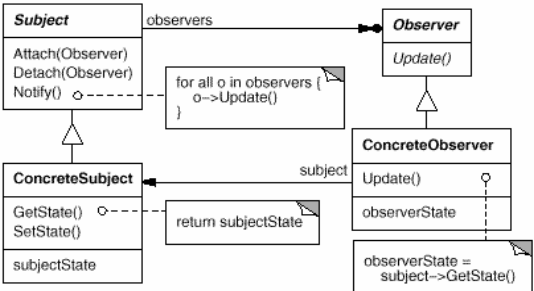
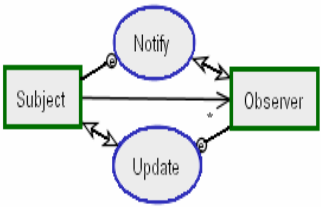
| OBSERVER  |  |
|---|--|
| <b>Problem Definition:</b><br>There is a one-to-many dependency relation between objects in the system. | <b>Suggested Solution:</b><br>Separate between the true object and its observers.                        |
| <b>UML Model:</b><br> | <b>OPM Model:</b><br> |
| From [11], p. 294.  |  |

Fig. 6. The Observer design pattern

characterizing pattern are Chain of Responsibility and Template Method. When modeling the Chain of Responsibility design pattern in OPM, for example, we get a structure of processes that is quite similar to the Decorator structure.

The OPM model of the Template Method design pattern uses the notation of an environmental process (dashed border lines). An environmental thing (object or process) in OPM is either external to the system (pattern) or is an abstract, under-specified thing that needs further specification in the target application model. The OPM model of the Template Method design pattern specifies that the "Template Method" consists of "Internal Operations", as well as "Primitive Operations" that should be further specified in the context of an application. This model is quite similar to the OPM model of the Factory Method; they both define functionality which should be embedded in a behavior

and can be preceded and/or followed by different operations. However, the focus of the Factory Method design pattern is behavioral—the embedded functionality creates products, while the emphasis of the Template Method design pattern is structural—the template consists of internal operations, as well as external (environmental) ones.

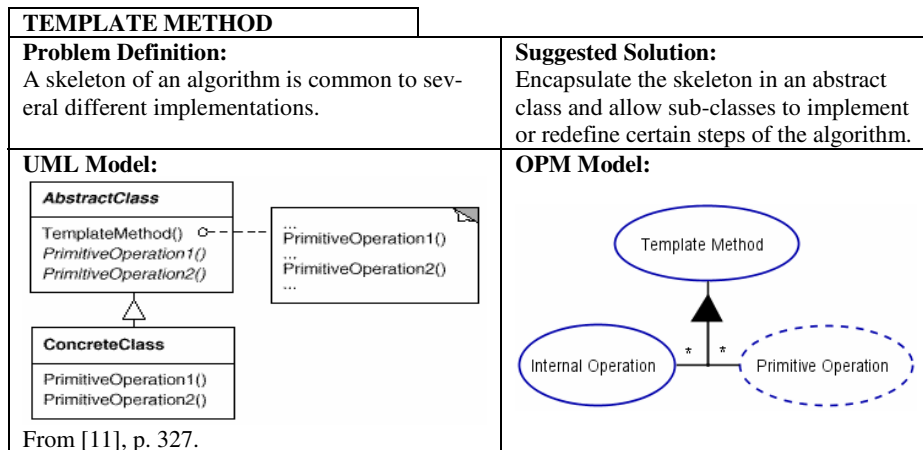


Fig. 7. The Template Method design pattern

#### 4.4 Classifying Design Patterns from an OPM Viewpoint

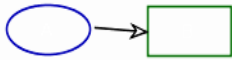
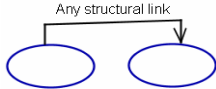
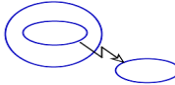
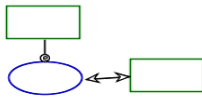
As we have seen, the OPM models of the design patterns reviewed in this paper induce a refined and precise way to classify design patterns. This classification includes four groups, listed in Table 2. The first group of *creational design patterns*, which is quite identical to the creational group in [11], is characterized by the OPM pattern of process – result link – object, which purely conveys the idea of a process creating an object.

The second group, *structural composition design patterns*, has the most abstract characterizing structure: two processes are connected via a structural relation of any type. In the Composite design pattern, for example, the in-zooming of "Composite" into "Component" reveals an aggregation relationship. Note that in the case of Chain of Responsibility, one process, "Handler", plays the role of both processes. Furthermore, the Chain of Responsibility design pattern is often used together with the Composite design pattern. As their OPM models show, these design patterns are very similar and belong to the same category.

The patterns in the third group are classified as *wrapper design patterns*, since they solve their stated problems by wrapping the original functionality.

Finally, the fourth group is the *interaction design patterns* group. The patterns in this group focus on the interaction between static and dynamic aspects of the solution. In OPM terms, these patterns emphasize procedural links, namely effect and event links, which are responsible for updating objects and triggering processes, respectively. This common structure can be observed in different manifestations in the OPM models of the different design patterns that belong to this group. In the OPM model of

**Table 2.** Classification of Design Patterns according to OPM models

| Design Pattern Category | Design Pattern Examples                                 | Typical OPM Model Core  |
|-------------------------|---|---|
| Creational              | Factory Method<br>Builder                               |  |
| Structural composition  | Chain of Responsibility<br>Composite<br>Template Method |  |
| Wrapper                 | Decorator   |  |
| Interaction             | Observer  |  |

the Observer design pattern, the characteristic structure of the interaction design patterns appears twice.

## 5 Conclusions and Future Work

To encourage software engineers to employ design patterns throughout the entire software development process, the design patterns should be classified logically and represented formally, so their retrieval would be effective and their usage—correct. Since different stakeholders engaged in systems development are more likely to remember visual representations of ideas than textual ones, both UML and OPM suggest ways to model design patterns graphically.

In this paper, we have presented OPM models of seven popular design patterns. We pointed out how the OPM models of the design patterns convey the essence of the solutions offered by the patterns and how these OPM models help designers integrate them into their application models. Comparing the design patterns' OPM models to their UML counterparts, we have shown that the former are more expressive and formal. Furthermore, we found out that the OPM models induce a logical classification of the design patterns into four groups: creational, structural composition, wrapper, and interaction. This classification refines the classification in [11] and identifies some problems in the categorization there (Chain of Responsibility and Template Method, for example, should be categorized as structural design patterns rather than behavioral ones).

We plan to apply our approach to additional design patterns and develop models of complete applications that host design patterns. We also plan to develop ways to retrieve design patterns easily (using OPM models) from the problem domain.

## References

1. Alexander, C., Ishkawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., Angel, S.: A Pattern Language. New York: Oxford University Press (1977).
2. Buschmann, F., Meunier, R.: A System of Patterns. In: Coplien, J. O. and Schmidt, D. C. (eds.): Pattern Language for Program Design, Addison-Wesely (1995), pp. 325-343.
3. Buschmann, F., Meunier, R., Rohnert, H., Sommerland, P., Stal, M.: Pattern-Oriented Software Architecture: a System of Patterns. Wiley (1996).
4. Chandrasekaran, B.: Generic tasks in knowledge-based reasoning: high-level building blocks for expert system design. *IEEE Expert* (1986), pp. 23-30.
5. Dietrich, J., Elger, C.: A Formal Description of Design Patterns using OWL. Proceedings of the 2005 Australian software engineering conference (2005), pp. 243-250.
6. Dori, D.: Object-Process Methodology – A Holistic System Paradigm. Springer (2002).
7. Dori, D.: ViSWeb – The Visual Semantic Web: Unifying Human and Machine Knowledge Representations with Object-Process Methodology. The International Journal on Very Large Data Bases, 13, 2, pp. 120-147, 2004.
8. Eden, A. H., Hirshfeld, Y., Yehudai, A.: LePUS – A declarative pattern specification language. Technical report 326/98, department of computer science, Tel Aviv University (1998).
9. France, R. B., Kim, D. K., Ghosh, S., Song, E.: A UML-Based Pattern Specification Technique. *IEEE Transactions on Software Engineering* (2004), 30 (3), pp. 193-206.
10. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Abstraction and Reuse of Object Oriented Design. Proceedings of the 7<sup>th</sup> European Conference on Object Oriented Programming. Berlin: Springer-Verlag (2003), pp. 406-431.
11. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley (1994).
12. Lauder, A., Kent, S.: Precise Visual Specification of Design Patterns. Lecture Notes in Computer Science (1998), Vol. 1445, pp 114-136.
13. Noble, J.: Classifying relationships between Object-Oriented Design Patterns. Proceedings of the Australian Software Engineering Conference (1998), pp. 98-108.
14. Object Management Group. UML 2.0 Superstructure FTF convenience document. <http://www.omg.org/docs/ptc/04-10-02.zip>.
15. Peleg, M., Dori, D.: The Model Multiplicity Problem: Experimenting with Real-Time Specification Methods, *IEEE Transaction on Software Engineering* (2000), 26 (8), pp. 742-759.
16. Petri, D., Csertan, G.: Design Pattern Matching. *Periodica Polytechnica Ser. El. Eng* (2003). Vol. 46, no. 3-4, pp. 205-212.
17. Reinhartz-Berger, I.: Conceptual Modeling of Structure and Behavior with UML – The Top Level Object-Oriented Framework (TLOOF) Approach. Proceedings of the 24th International Conference on Conceptual Modeling (2005), Lecture Notes in Computer Science 3716, pp. 1-15.
18. Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorenson, W.: *Object-Oriented Modeling and Design*. Prentice Hall, Englewood Cliffs, NJ (1991).
19. Shalloway, A., Trott, J.: Design Patterns Explained: A New Perspective on Object-Oriented Design. Addison-Wesley (2001).
20. Zimmer, W.: Relationships Between Design Patterns. In: Coplien, J. O. and Schmidt, D. C. (eds.): Pattern Language for Program Design. Addison-Wesely (1995), pp. 345-364.

# Modeling and Simulation of Grid Information Service<sup>\*</sup>

Xia Xie, Hai Jin, Jin Huang, and Qin Zhang

Cluster and Grid Computing Lab.,  
Huazhong University of Science and Technology, Wuhan, 430074, China  
hjin@hust.edu.cn

**Abstract.** Grid information service is an important part of grid systems, which influences performance of grid systems directly. Therefore, it is necessary and urgent to analyze characteristics of grid information service. We analyze performance metrics of grid information service; discuss modeling of grid information and grid information service. As an example, JFreeSim is introduced and focused on simulation of grid information service. JFreeSim realizes many entity modeling and communication mechanisms among all entities, and makes system simulation in accord with the characteristic of the grid environment. Examples indicate that JFreeSim can provide users with flexibility in configuring and satisfying requirements of different information service modeling.

## 1 Introduction

Flexible, secure and coordinated resource sharing between *virtual organizations* (VOs) requires the availability of an information rich environment to support resource discovery and decision making processes. The discovery, characterization, and monitoring of resources, services, and computations can be a challenge due to the considerable diversity, large numbers, dynamic behavior, and geographical distribution of the entities in which a user might be interested. Thus, information services are an important part of any grid platforms, providing fundamental support for discovery, gathering and monitoring, and thus for planning and adapting application behavior. The *grid information service* (GIS) must be able to gather and manage ‘dynamic’ data – data that has often short lifetime of utility, or which is updated frequently.

Many new ideas or new technologies of GIS are explored. In order to compare and analyze the performance, the best way is to test them with real application in real grid environment. But GIS is related to grid platform closely. Performance of GIS is important to estimate performance of whole grid platform. On the contrary, influences from grid platform to GIS is recessively or well-marked, such as dynamic network status. Moreover, it is expensive and difficult to build a real grid environment just for testing. Therefore, modeling or simulation is often used. Modeling and simulation can help us get performance characteristics by analysis or in a virtual environment. In this

---

<sup>\*</sup> This paper is supported by ChinaGrid project funded by Ministry of Education of China, National Science Foundation under grant 90412010, and China CNGI project under grant CNGI-04-15-7A.

paper, we introduce JFreeSim, a new grid simulation tool based on multiple tasks, multiple schedulers and multiple resources (MTMSMR) model. Experiments indicate that JFreeSim can provide users with flexibility to evaluate performance of GIS and the simulation results are as expected.

The rest of this paper is structured as follows. We introduce related work in section 2. In section 3, we analyze performance metrics of grid information services. In section 4, we discuss modeling of grid information and modeling of grid information service. We design a GIS simulation tool, JFreeSim, in section 5. Finally, we draw conclusions in section 6.

## 2 Related Works

The most relevant systems are Globus MDS [4], Condor Matchmaker [7], and R-GMA [6]. MDS approach to construct GIS is to push all information into a directory service. While this system pioneered information services for the grid, the strategy of collecting all information into a database inevitably limited scalability and reliability. In Condor Matchmaker, both resources and queries are collections of attributes and constraints. This enables matchmaking, where both the resource owner and the requester can constrain which results are returned on a query. R-GMA is a system, which uses a relational grid monitoring architecture.

Performance experiments on MDS2 and R-GMA [10] finds a strong advantage caching or pre-fetching the data, as well as the need to have primary components at well-connected sites because of the high load seen by all system.

Since it is difficult to evaluate performance of GIS in real environment, several kinds of simulations are introduced. Simulation tools for parallel and distributed systems are listed but not limited as follows: GridSim [1], SimGrid [2], MicroGrid [8], and etc. MicroGrid supports scalable simulation of grid applications using clustered resources in computational grid research and can deploy real grid applications on it. SimGrid simulates distributed applications in heterogeneous distributed environments. GridSim enables modeling and simulation of entities involved in parallel and distributed computing through the creation and monitoring of different resources. They are library-based and focus on resource scheduling algorithms [9] in a grid computing environment. Modeling framework of each tool is entity-based or event-based. But none of them focus on grid information service directly. That is, none of them provide relative mechanism or modules to simulate grid information service. None of them support or provide interfaces for simulation of new grid information service modeling. Most of them can give trace data or charts after each running but less of them can give a precious statistic and analysis, which can be shown the detailed performance of different grid information service.

## 3 Metrics for Grid Information Services

Information system needs to present the data with semantics that afford rich expressiveness and flexibility, it must also be able to gather and deliver the data in an efficient manner. Moreover, it is necessary to present the data in multiple formats or



encodings. The format translations must also be sensitive to overhead and not impose significant performance penalty.

The main challenge of GIS is where to locate the right information to do the right thing. This issue must be resolved from two sides. One is from grid information services. It includes proper index updating algorithm, index service policies, information collection mechanism, type of queries and etc. Type of queries includes not only the direct queries but also queries with some special characteristics, such as time limitation or reliability-constraint. Another is from grid resource. Grid resources need to obey some agreements negotiated by GIS and grid resource to register in GIS.

For performance of GIS, both users and designers are concerned whether the information service can provide proper information and finish tasks with high efficiency. Usually, grid system provides a single interface or portal for user, such as a web portal. User browses from the web portal and submits his queries or tasks. The grid platform will search related information to match queries and then implement tasks. The large-scale heterogeneous grid is subject to frequent changes and services disruptions of some of the huge number of components that constitute this environment. This also influences the performance of GIS. GIS includes several parts, such as information discovery, information gathering. Efficient design and implementation can decrease system cost. Otherwise, it will be the bottleneck of grid system.

The resource requirements of users can be satisfied in many ways. Information broker establishes a proper mapping between the specified resource needs and the available physical resources. The performance is substantially determined how appropriate resources are discovered, gathered and monitored. It must be made clearly whether it generates specifics, available information set; whether it spends the least time; whether it adds new burden to the whole system (e.g. send vast sockets); whether the information set generated is effective and so on. Performance of a grid application has many aspects and it depends on many possible parameters like CPU power, network speed, cache efficiency, scheduling policies, communication scheme. For GIS, user always concerns on whether needed data or files could be found in complex environment before deadline; whether information set is enough abundant and exact; whether monitoring requirement brings new system cost.

The general metrics for GIS are listed below:

- Throughput. It is defined as the average number of requests processed by information service mechanism per second.
- *Average response time* (ART). It is the average time span (in seconds) from user's request sent out to all the responses returning from information service received by the user.
- CPU\_Load. It is the average number of processes in the ready queue waiting for run over the last minute.
- Successful\_Rrequest (SR). It indicates the percentage of the successful request processed in all of the requests.
- Resource\_Usage. It is the percentage used part for a kind of resources in grid system.

We mainly focus on the first two metrics for GIS performance in this paper.

## 4 Modeling

### 4.1 Modeling of Grid Information

Since our work focuses on how to design a grid simulator and use the simulator as a testbed to evaluate the performance of information service mechanism, we specify general types of information that are vital in a simulation environment as follows:

- Task processing status: it is used to tell user the status of his task. It can be composed with running, ready for scheduling, waiting for resources, suspending for errors and etc. Each user can monitor his tasks according to task processing status.
- Grid resources: they are physical resources in grid system or virtual resources in grid simulator, such as CPU speed, memory capacity, files, data, cluster, and workstation. All the tasks need to utilize particular grid resources by user's requirement. Grid resource is a complex data structure, which includes resource type, size, and access, permitted operations and so on.
- Information index: it is the high-level abstract that describes a particular data set and management organization that can access the data. Information index also has its own structure. Several information indexes in same layer can be managed or controlled by other information index in higher layers. Generally, user's query is passed from the highest layer to lowest layer step by step according to lookup or discovery algorithms. Sometimes user's query can be delivered to appropriate layer directly by policies.
- Queue status: it is used to demonstrate the jobs, which are waiting for scheduling and resource allocation in a grid system or in grid simulator.

A grid simulation needs to represent real grid environment and provide test conditions for user's comparison and analysis. All of the above need to be implemented in grid simulator. Representing information in GIS is often based on metadata and supported by LDAP, UDDI and WSIL. Different grid platform has different information representation format. Sometimes, interaction model based on relational database is used. Format of metadata is defined by grid system developer and must obey OGSA infrastructure.

### 4.2 Modeling of Grid Information Service

Information service is a kind of grid service with distributed structure or P2P structure. The later is much more suitable for large-scale and range discovery in grid environment. Modeling of grid information service needs to be easy to build GIS mechanism. Generally, grid information service needs four basic components: index service, information provider, users and resource brokers.

Information providers get dynamic or static information from multiple resource brokers. Index service provides registration service like UDDI with better scalability. Index service gets and aggregates resource information from information providers, releases the information and provides resource discovery service for users. Index service can register to each other so that a distributed information service is formed. All the information can be gathered into different levels and then user can submit resource discovery request to index service in higher level directly. In grid

environment, information has its own lifecycle. If it expires before updating, it will be removed from index service.

In order to simulate GIS mechanism efficiently, after analyzing the existing GIS systems in detail, we design the three independent layers to simulate GIS in JFreeSim as Fig. 1.

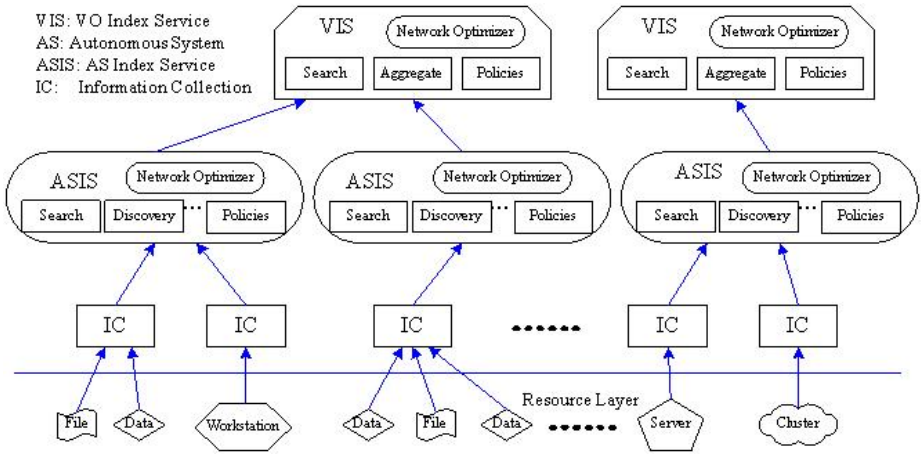


Fig. 1. Structure of GIS in JFreeSim

- *VO Index Service (VIS)*. It is user-oriented module and can support aggregating, querying, discovering and allocating the resources for the user, and offer other functions according to the needs of grid system. In addition, VIS may get resource information subscription from AS Information Service. VIS may also send queries to AS Information Service module to obtain the resource information according to special policies. VIS also can register to each other.
- *AS Information Service (ASIS)*. Like *Autonomous System (AS)* in OSPF, AS in grid is a sub-network in VO. ASIS serves mainly for all users in the sub-network. It classifies and combines the information from registered information collection module, and organizes for VO information service. Its functions include lookup, query, and monitoring. Relative policies are needed. One or more ASIS consist of a VO information service. ASIS also can register to each other.
- *Information Collection (IC)*. It is used to collect real information of grid resources and other services, such as status of the object monitored. It collects information from resource brokers, and obtains static or dynamic data about resources. One or more information collection modules can register to an ASIS. IC also can register to each other.

Network Optimizer is optional component. It is used to sort candidate resources by network distance in decreasing order. Network distance is the distance between user and resource node. We often use network hops or network latency to present it. By these three modules, people can build various information service mechanism. JFreeSim provides various configure options, including the instance of function modules used, the connection of modules, the deployment of resources and applications.

5 JFreeSim: A Toolkit for Modeling and Simulation of Grid Information Service

We design a simulation tool, named JFreeSim, to study the performance of GIS with Java. Using this simulation toolkit, we can simulate a range of grid environments. Design environment for JFreeSim is based on SimJava. There are seven entities defined in JFreeSim. Modeling of GIS is implemented to build various kinds of information service mechanism, such as for large-scale and range discovery. JFreeSim has friendly graph-interface and can integrate and analyze results in table style or chart style. Each module can be recoded by researcher’s modeling. Besides this, it provides a dynamic presentation about the whole simulation flow and analyses metrics. Fig.2 describes the architecture for JFreeSim.

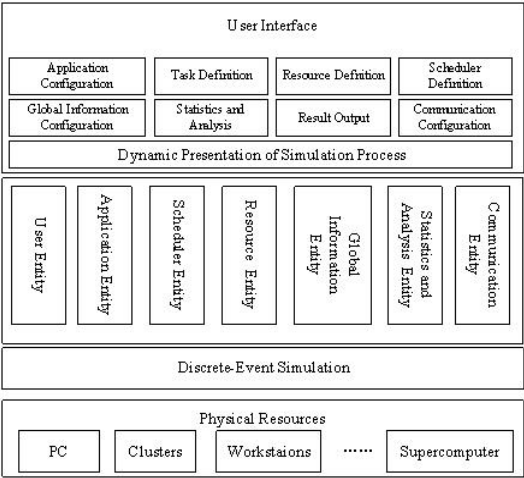


Fig. 2. Architecture of JFreeSim

5.1 System Architecture

JFreeSim has four levels. The first level is user interface. It includes nine parts: application configuration, global information configuration, resource definition, statistics and analysis. This level is used to submit user’s requirement, choose parameters, calculate and analyze simulation data, show the results. For example, a function of communication configuration is to allocate IP address for each virtual node.

The second level is key part of whole architecture and composed with seven entities. An entity has representation in a grid service as an object. An object is a representation of a real-world entity. Objects are often described by a set of value-attribute pairs. Global information entity focuses on GIS simulation. User entity is responsible for creating and initializing other entities according to the user’s global configuration. In addition, user instance creates application and task instances according to the system configuration as well. Each computing node receives tasks from the scheduler, executes the tasks according to its local scheduling policy, and reports updating

performance information to the information directory. Communication entity supports data receiving and transmission for discovery and query. UDP is used when communication is needed between information service nodes. Virtual IP address is used for receiving and transmission of data message without real IP address. The simulator displays various build-in and user-defined measurements with the comprehensive graphical interface through the special statistics entity. JFreeSim is also able to analyze the activities of entities for simulation and determine possible performance bottleneck according to user's modeling or requirement.

The third level is the discrete-event simulation based on SimJava, which uses object-oriented programming. SimJava models the simulation objects as entities that communicate by sending passive event objects to other entities via ports. Detailed statistics generation is provided in the SimJava library so that the user can indicate which simulation events are collected and written into a trace file [5].

The last level is the physical resources, including PCs, clusters, workstations. This part provides logistic processing elements. All the tasks are implemented on these elements. The last level is composed with virtual physical resources.

## 5.2 Scalability

There are two major factors affecting the scalability of a grid system. First, scalability depends on available parallelism, and thus, whether and how to find large parallelism for a given network simulation is a key issue. There is significant on-going research on parallel and distributed network simulation. Second, scalability depends on the achieved load balance. Load balance is important because it directly affects the simulation performance: the speed of the parallel/distributed simulation is limited by the speed of the node with the most loads.

Scalability is also a key problem to simulate GIS mechanism. When we simulate, both single CPU parallel availability and large-scale parallel network simulation may affect scalability of GIS. For example, SimGrid and GridSim are not suitable for high scalability GIS simulation. In order to avoid these two problems, we use multiple PCs for simulation; use Inet [3] to generate a large-scale network topology; use ModelNet [11] to design link characteristic of this network topology. ModelNet maps a user-specified topology to a set of core routers that accurately emulate on a per-packet basis the characteristics of that topology, including per-link bandwidth, latency, loss rates, congestion, and queuing. For resource discovery and query, virtual nodes exchange communication messages on the network generated by ModelNet.

JFreeSim is based on SimJava. If entities are in a same virtual node, communication between entities is implemented by `sim_port` message. If entities in different nodes need communicate with each other, communication message is transmitted in large-scale network generated by Inet and ModelNet. For example, information subscription, information registration, discovery and query in GIS are deployed. It is same for data or files transmitted between user and resource providers for sharing and cooperation.

If resource discovery and message query use TCP, it will has more strictly or higher requirement to system. For instance, maintaining information service connection among multiple virtual nodes is easy to reach or break upper limitation of system memory or operation system threads. Then UDP is used to support communication between information service nodes.

### 5.3 Procedure of Building Simulation with JFreeSim

In JFreeSim, procedure of application simulation is as follows:

- First, user submits requirements from user interface. Each entity is initialized and the whole simulation procedure starts. Global information entity receives registration from all the available resource brokers through UDP on the simulated network by Modelnet.
- Second, application entity sends user's information in turn to scheduler entity with SimJava's message mechanism.
- Third, the scheduler queries the global information directory.
- Fourth, when a user sends queries to web portal, web portal transmits queries to GIS mechanism. The first VIS or ASIS that receives queries is called local VIS or local ASIS. Local VIS or local ASIS will try to match queries in its own resource directory. If not matched, queries are transmitted to other VIS or ASIS by policies.
- Fifth, local ASIS or VIS sorts list of resources matching query condition according to network distance between user and each candidate resource nodes and return query response to user.
- Last, the scheduler assigns tasks to suitable resource broker by special scheduling policy default or defined by researchers.

In the first step, these requirements are configuration parameters. Parameters can show how many users created; how many kinds of grid resources are built; what the main characteristics for each kind are; how to build a network topology using JFreeSim to simulate network (including routers, switches, hubs, link status, network path); give some original data to start Inet and Modelnet to build a simulated network. Besides this, script file is also accepted by user interface if the file follows the definition of JFreeSim.

Every resource maintains a local task queue and simulates the processing of tasks. When a task is over, the status of the task will be updated in the global information entity. The dynamic characteristic also can be simulated in JFreeSim. In the whole simulation flow, some resources may be available but others fail to access. After a while, resources can change their status. All of these can be traced in global information entity. For the updating frequency of global information entity, it can be fixed or event-driven. Moreover, each modular in JFreeSim can be replaced according to user's research interest. Every part is independently modeled, and interacts by a certain message mechanism. The extensible design allows the user to construct different simulation script for various applications and execution process.

### 5.4 Case Study

JFreeSim can help us to simulate and evaluate new mechanism of GIS. For example, in order to support network-sensitive grid applications well, we model and simulate a kind of GIS based on network latency. Through the network latency, GIS can calculate the network distance between user and resource providers. Network latency is an attribute provided for resource discovery in this kind of GIS, which can not only use network latency as a query characteristic but also can return a list of candidate resource providers with latency in increasing order automatically. In order to predict the network latency between user and candidate resource providers in grid system, network coordinates are

introduced to describe the location of each grid node. Information service sorts and returns a list of candidate resource providers to users with network latency increasing. Then user can choose proper resource provider with minimum network latency to use, which can accelerate the running of applications.

The test environment is built as follows. A cluster consists of 16 PCs with 1GHz Pentium III processors and 512 MB SDRAM with FreeBSD 4.5 operating system. We use the Inet topology generator to create 8,000-node wide-area AS-level network with a variable number of 500 client nodes always with 4 client nodes per stub. ModelNet is used to emulate wide-area bandwidth and latencies. Transit-transit and transit-stub links are 155 Mb/s and client-stub links are 1Mb/s. Latencies are based on the Inet topology. The 500 client nodes are simulated by 11 PCs running JFreeSim. Each node can send or receive data in simulated network environment and each data packet is embedded IP address of virtual node, which does not use the real NIC address of 11 PCs. Both latency calculation algorithm is implemented in each information entity, grid user entity or resource entity generated by JFreeSim. Each PC has 1.6 GHz Pentium IV processor and 512 MB DDR RAM with Redhat 9.0. Both cluster and PC use Gigabit Ethernet. The cluster is used to simulate network environment and PCs are used to simulate grid environment. Each PC runs JFreeSim. We select one of 11 PCs to simulate information service and the remaining is used to simulate user nodes or resource node. Configuration parameters for each virtual node are input from special configuration files by JFreeSim. Direct queries are used in these tests. Network topology is dynamically changed by the configuration of Inet and ModelNet.

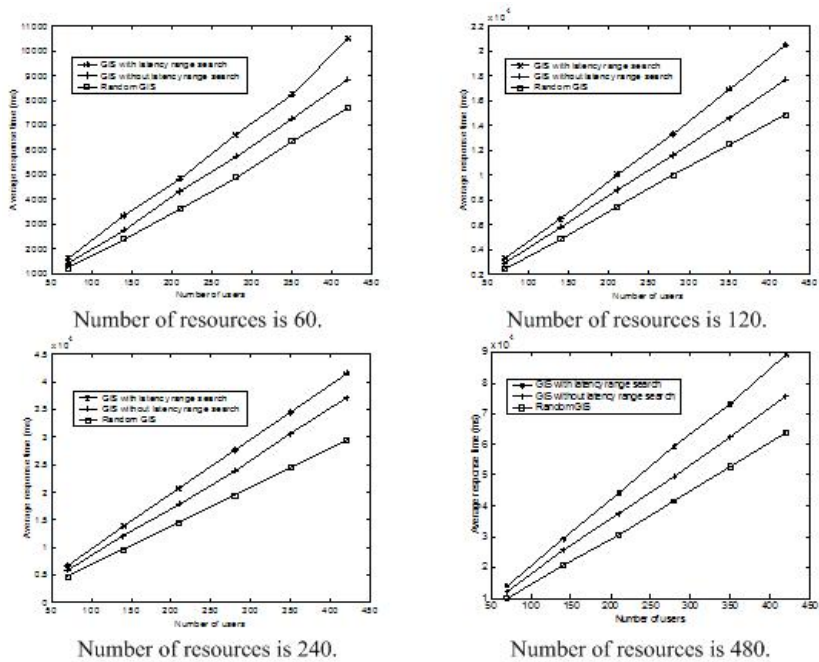


Fig. 3. Average Response Time of Information Discovery with Number of Resources

Two metrics are used here: 1) *average response time* (ART) of information discovery; 2) throughput of information service. Average response time of information discovery is the time zone from information discovery requirement sent out by user to results returned. It is an average value for multi-times simulations. In order to compare freely, we call the general information service without network latency optimization “random information service”. This section will show the comparison with: 1) GIS with latency range search, which regards network latency as a condition for information discovery and optimize it in GIS; 2) GIS without latency range search, which just provides network latency optimization; 3) GIS with random information services, which is only MDS4 and does not care network latency. We call them “type A, B, C”, respectively.

Fig. 3 describes the average response time of information discovery for A (latency<120ms), B and C, when the number of resources is 60,120,240,480, respectively. Comparing these figures, we can draw conclusions as follows:

- Compared with type C, type A and B have longer average response time of information discovery.
- Type A has longer average response time of information discovery than type B.
- More resource providers, more average response time of information discovery. The limitation of scalability is for MDS4 itself.

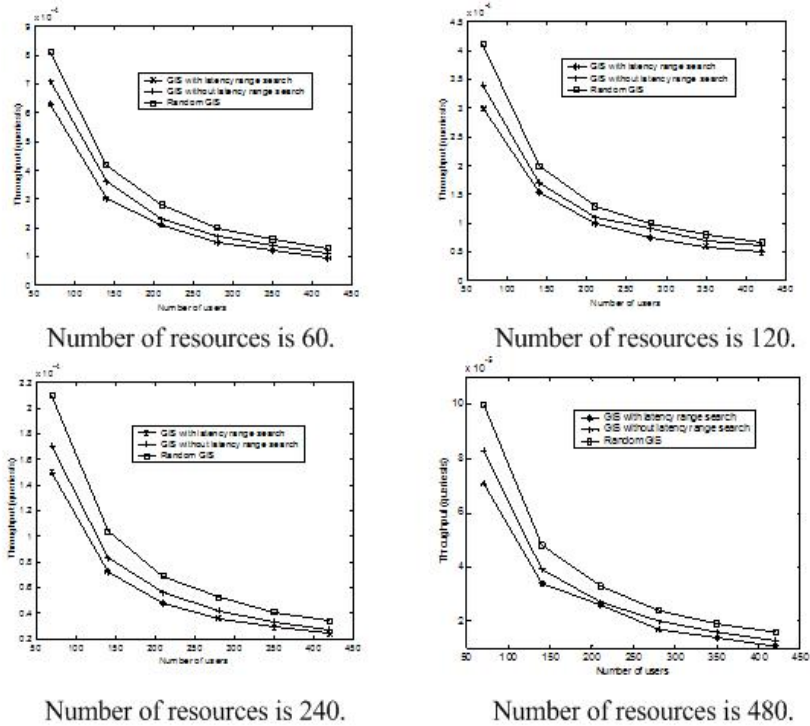


Fig. 4. Throughput of Index Services with Number of Resources



Fig. 4 describes the throughput of index service for A (latency<120ms), B and C, when the number of resources is 60,120,240,480, respectively. We find:

- Compared with type C, type A and B has smaller throughput of index service.
- Type A has smaller throughput of index service than type B.
- More resource providers, less throughput of index service.

## 6 Conclusion and Future Work

Performance evaluation of GIS is very difficult in real environment. Information sources are necessarily distributed and individual sources are subject to fail. The total number of information provided can be large, and both the types of information sources and the ways information used can be highly varied. This paper dedicates to present performance metrics of GIS, introduces modeling of grid information and GIS. Based on these, JFreeSim is introduced as a toolkit for modeling and simulation of grid information service. JFreeSim provides simulated grid environment so that designer can control all the tasks and resources available to evaluate a new GIS mechanism, such as reliability-constraint.

Our future work is to simulate service-oriented GIS based on JFreeSim. New policies and technologies of GIS will also be simulated.

## References

1. Buyya R. and Murshed M., "GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing", *Concurrency and Computation: Practice and Experience*, Wiley Press, 14, 2002, pp.1175–1120.
2. Casanova H., "SimGrid: a Toolkit for the Simulation of Application Scheduling", *Proceedings of the First IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid'01)*, May 15-18, Brisbane, Australia, 2001, pp.430–437.
3. Chang H., Govindan R., Jamin S., Shenker S., and Willinger W., "Towards Capturing Representative AS-level Internet topologies", *Int. J. Computer and Telecommunications Networking*, Vol.44, 6, 2004, pp.737–755.
4. Czajkowski K., Fitzgerald S., Foster I., and Kesselman C., "Grid Information Services for Distributed Resource Sharing", *Proceeding of 10th IEEE International Symposium on High-Performance Distributed Computing (HPDC-10)*, 2001, pp.181–194.
5. Howell F. and McNab R., "SimJava: A Discrete Event Simulation Package for Java with Applications in Computer Systems Modeling", *Proceedings of First international Conference on Web-based Modeling and Simulation*, San Diego, CA, 1998, pp.55–64.
6. Podhorszki N. and Kacsuk P., "Monitoring Message Passing Applications in the Grid with GRM and R-GMA", *Proceedings of EuroPVM/MPI 2003*, Venice, Italy, 2003, pp.603–610.
7. Raman R., Livny M., and Solomon M., "Matchmaking: Distributed Resource Management for High Throughput Computing", *Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing (HPDC'98)*, 1998, pp.140–146.
8. Song H. J., Liu X., Jakobsen D., Bhagwan R., Zhang X., Taura K., and Chien A., "The MicroGrid: a Scientific Tool for Modeling Computational Grids", *Proceedings of IEEE Supercomputing (SC2000)*, Nov. 4-10, Dallas, USA, 2002, pp.127–141.

9. Takefusa A., Matsuoka S., and Nakada H., "Overview of a Performance Evaluation System for Global Computing Scheduling Algorithms", *Proceedings of 8th IEEE International Symposium on High Performance Distributed Computing (HPDC-8)*, 1999, pp.97–104.
10. Xuehai Z., Jeffrey L. F., and Jennifer M. S., "A Performance Study of Monitoring and Information Services for Distributed Systems", *Proceedings of 12th IEEE International Symposium on High Performance Distributed Computing (HPDC'03)*, 2003, pp.270–281.
11. Vahdat A., Yocum K., Walsh K., Mahadevan P., Kostic D., Chase J., and Becker D., "Scalability and Accuracy in a Large-scale Network Emulator", *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI'02)*, 2002, pp.271–284.

# Simulations of Distributed Service-Based Content Adaptation for Network Optimization

Eric Y. Cheng<sup>1</sup> and Shuo Hung Jian<sup>2</sup>

Department of Information Management, National Central University  
320 Chung-Li, Taiwan

<sup>1</sup> ycheng@mgmt.ncu.edu.tw

<sup>2</sup> 93423029@cc.ncu.edu.tw

**Abstract.** The most traditional approach to network optimization focuses on bringing new or improved capabilities to targeted functional areas of network operations. The goal is to boost the unit's efficiency and effectiveness. There are three different approaches proposed to help in designing adaptive multimedia systems for heterogeneous devices: server-side adaptation, proxy-based adaptation, and adaptation paths. Though there are a variety of approaches being proposed, evaluating content adaptation approaches using simulation is still lacking. This paper reports a simulation effort to compare the differences of performance among those three kinds of content adaptation approaches. A 2 by 5 factorial design was implemented to test the effects of five influential factors within the network. The result indicates that the service based approach is not only capable of adapting varied network environments but also have outperformed the other two. This could be used as a key reference for the optimization of content adaptation network.

**Keywords:** Content adaptation, network optimization, simulation, factorial design, operations research.

## 1 Introduction

Pervasive computing or ubiquitous computing is the trend towards increasingly ubiquitous, connected computing devices in the environment, a trend being brought about by a convergence of advanced electronic - and particularly, wireless - technologies and the Internet.

For pervasive computing to succeed in general, fundamental issues in different areas such as network, data management and delivery and security among others, have to be well addressed [1]. Heterogeneity and mobility of devices poses new challenges for information delivery applications in this environment. To meet the demands in this heterogeneous environment, it is necessary for the information to be customized or tailored according to the user's preferences, client capabilities and network characteristics [8]. This tailoring process is called content adaptation which may include format trans-coding (e.g. XML to WML, JPEG to WBMP), scaling of images as well as video and audio streams, media conversion (e.g. text-to-speech),

omission or substitution of document parts (e.g. substitution of an image by a textual representation), document fragmentation, language translation, etc.

On one hand, Web Services are becoming popular technologies for publishing various services on the Internet [21]. On the other hand, there is a trend of developing content adaptations as value-added services [2][18]. Dynamism and diversity of content adaptation mechanisms combined with the opportunities came to exist like Web Service technologies, open new approach of content adaptation which is service-based. In this approach adaptations are done by third-party services on the fly. These services can be accessed through subscription by users or content providers. To realize this approach at least the following things must be addressed well:

- There is a need of flexible and scalable architectural framework to incorporate easily third-party services.
- Depending on the complexity of the content and context of the environment, it may be necessary to apply multiple adaptation techniques on the data before it reaches the user (e.g. changing a text in Chinese to English, extract summary of the text and then translate the text to audio). Furthermore, one or more services could provide similar adaptation service, for example, video adaptation with different size, cost and quality values. In order to determine adaptation services suitable for the client or the environment context in general with some quality of services, a formal model of content negotiation and adaptation is important.

Based on the above reasons and for there are a variety of content adaptation approaches and their performances are unknown, there is a need for optimized content adaptation network. The purpose of this study is to compare the performance differences among service-based content adaptation approach and other existing approaches through simulation.

## 2 Related Researches

Early efforts date back to the DeleGate Gopher proxy [20] for Kanji transcoding that started in 1994. Leveraging content adaptation to adapt to the capabilities of mobile devices was investigated by the Daedalus project at the UC Berkeley. It has proven that on demand dynamic adaptation of text and images by proxies is feasible and powerful [11]. The concept of proxy-based dynamic adaptation was further refined by the UC Berkeley's Ninja Project [9]. The project proposes a robust, scalable architecture for web access by heterogeneous devices. In the Ninja project, on-demand dynamic adaptation is done in a step-by-step manner by several adaptation proxies establishing an adaptation path, the so-called Ninja Path. The ideas of Ninja Paths are the basis for the step-by-step adaptation used in our approach. Ninja assumes a centralized instance of the path subsystem implementing the path composition and implementation [4]. While the Ninja approach guarantees the composition of valid paths, path optimization is not addressed.

Algorithms for composing and optimizing a distributed adaptation path have been dealt with by [17][5]. Ooi et al. [17] describe an algorithm to distribute the computation of multimedia streams across multiple multimedia gateways. However, the algorithm presented in the paper optimizes the computation with respect to resource

consumption only. Aspects such as sharing of intermediary results are not addressed. Likewise, caching is not an issue. Kasim et al. [5] present a more general and more abstract approach. They describe algorithms to determine a valid distributed adaptation path to make a multimedia object in a certain representation available at the client node. The origin object may reside on different nodes in different representations. In [5], the authors describe bottom-up as well as top-down algorithms. Furthermore, they have proven the top-down algorithm to determine the optimal adaptation path. These algorithms, however, do not deal with caching or sharing of intermediary adaptation results. Even though, the bottom-up algorithm is the basis for the algorithm we are developing for the proposed architecture.

Furthermore, a lot of efforts have been spent on the adaptation of web documents. Automatic reauthoring of HTML pages to adapt them to the capabilities of mobile devices is done by the Digester system [3] and its successor, the m-link system [19], by Fuji Xerox Palo Alto Laboratory (FXPAL). Those systems use heuristics to understand web documents to enable content adaptation. However, the potentials of such heuristics are limited. In order to overcome those limitations, other work considers generic XMLbased representations of documents that are augmented by meta information to allow smarter adaptation [10]. The approach presented in this paper does not rely on the one or the other concept for the adaptation of web contents. It is meant to deal with arbitrary means of content adaptation.

The Open Pluggable Edge Services architecture (OPES) [18] work deals with standardizing mechanisms to extend intermediaries with application-specific value added services and provides a standard way for the content provider and/or third parties to offer services. This work focuses at general level and does not address in depth the issues of content adaptation.

The Internet Content Adaptation Protocol (iCAP) [7] distributes Internet-based content from the origin servers, via proxy caches (iCAP clients), to dedicated iCAP servers. For example, simple transformations of content can be performed near the edge of the network instead of requiring an updated copy of an object from an origin server, such as a different advertisement by a content provider every time the page is viewed. Moreover, it avoids proxy caches or origin servers to perform expensive operations by shipping the work off to other (iCAP) servers. However, it only defines a method for forwarding HTTP messages, that is, it has not support for other protocols and for streaming media (e.g. audio/video) and only covers the transaction semantics (how do I ask for adaptation?) and not the control policy (when am I supposed to ask for which adaptation from where?)

### 3 Design of Simulations

Simulation software has been used to speed up the multi-stage and iteration simulating processes. A well-known simulation software, ARENA, has been employed for such purpose. ARENA is a comprehensive tool designed to make building and solving efficiency difference among different models faster, easier, and more efficient. Additionally, ARENA also provides a full-featured environment for building and editing problems, and a set of fast built-in solvers. Details of simulation design will be addressed in the following sections.

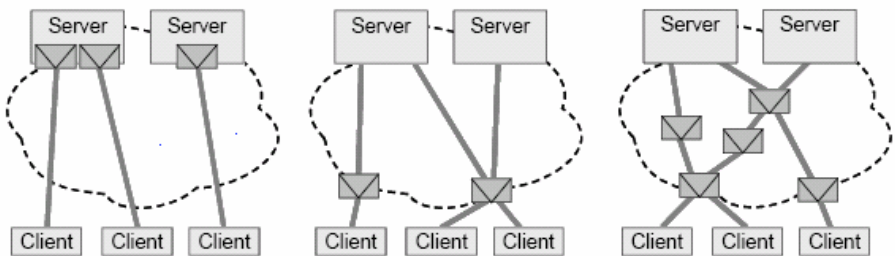
### 3.1 Content Adaptation Approaches

Most of the solutions and architectures proposed to help in designing adaptive multimedia systems for heterogeneous devices are mainly classified to three different approaches: server-side adaptation, proxy-based adaptation, and adaptation paths (we call it service-based adaptation in this paper).

Server-side adaptation approach [15][16], the functionality of the traditional server is extended by adding content adaptation. In this approach both static(off-line) and dynamic (on-the-fly) content adaptation can be applied and better adaptation results could be achieved as it is close to the content; however clients experience performance degradation due to additional computational load and resource consumption on the server [12] (cf. figure 1a).

Proxy-based adaptation [12][13], a proxy that is between the client and the server, acts as a transcoder for clients with similar network or device constraints. The proxy makes request to the server on behalf of the client, intercepts the reply from the server, decides on and performs the adaptation, and then sends the transformed content back to the client. In this approach there is no need of changing the existing clients and servers. The problem of proxy-based adaptation approaches is most of them focus on particular type of adaptation such as image transcoding, HTML to WML conversion, etc. and they are application specific. In addition, if all adaptations are done at the proxy it results in computational overload as some adaptations are computational intensive and this degrades the performance of information delivery like the server-side approach. (cf. figure 1b)

Service-based approach [6] is predicated on the finding that adaptation is often composed of elementary, successively applied adaptation steps. Just as with proxy-based adaptation, the server provides the document in a generic representation. On its way to the client, the document passes several distributed adaptation proxies implementing the adaptation in a step-by-step manner (cf. figure 1c). That way a distributed adaptation path is established.



a) Server side adaptation      b) Proxy based adaptation      c) Service based adaptation

**Fig. 1.** Approaches for Implementing Content Adaptation

### 3.2 2 by 5 Factorial Design and Settings

Because our goal is to compare the performance difference among three kinds of content adaptation approaches, for the comparison must be fair, each content adaptation

approach has to be run under the same environment. In addition, we assume that every server providing adaptation service is M/M/1 queuing system and each one is FIFO queue. Each content adaptation request all yields to Poisson distribution, each process time that server performs content adaptation all yields to Exponential distribution and each server have only one service window. Those assumptions mentioned above are close to real world conditions.

There are five factors have been defined and studied that would affect the performance of content adaptation. Additionally, a total of 32 scenarios have been designed through combination of varying five factors and two different levels. The following are the definitions of those five influential factors for content adaptation processes:

- Content Adaptation Request Rate: How many clients send content adaptation request per minute.
- Number of Adaptation Sub-process: How many adaptation sub-processes need to be performed in order to meet a content adaptation request.
- Process Time of Adaptation Sub-process: How much time is spent in implementing a content adaptation sub-process.
- Network Latency Time: How much time is spent in transmitting content. The value will vary depending on size of content. The larger content is, the higher network latency time. Additionally, it is also affected by whole network environment. For example, network latency time will be higher obviously at peak hours than at non-peak hours.
- Size of Buffer: How many content adaptation requests can be cached in server in busy. The higher it is the less content adaptation request is lost.

A 2 by 5 factorial design has been adopted to implement this experiment. For each factor there are two different levels, high and low, to represent peak non-peak conditions within the network. There are five factors have been examined in the experiment, which using one factor-at-a-time design [14]. Since there is no previous study available that could characterize the fundamental assumptions of content adaptation, we used simplified assumptive values in this experiment. Although the value is simple, it still provides valuable insights into the performance of content adaptation approaches. Nevertheless, using more realistic values to evaluate each content adaptation approach will be in the next stage as future work. First, we assume content adaptation request rate for either 1 request/min (peak hours) or 3 request/min (non-peak hours). Secondly, we assume the number of adaptation sub-process for either 2 sub-processes (simple adaptation) or 5 sub-processes (complex adaptation). Thirdly, we converted WORD .doc file into .pdf file using Adobe Acrobat 7.0 and calculated transformation time (run in the IBM server whose CPU is 3G Hz and Memory is 1 GB) to be the processing time. The process time is either 25 seconds (transcoding 100 KB .doc file into .pdf file) or 53 seconds (transcoding 1MB .doc file into .pdf file). Next, we calculated the average value from page loading time (download 100 KB file from server) of one hundred different websites all over the world from [www.internettrafficreport.com](http://www.internettrafficreport.com) and used the average value to be the network latency time. The network latency time is either 0.74 seconds (we can regard it as transmitting 100 KB file in the network or transmitting a file at non-peak hours) or 7.4 seconds (assume that the time of downloading 1MB file is 10 times than downloading 100 KB file or transmitting a file at peak hours). Lastly, we set size of buffer to be either 10

quantities (a small size of memory) or 30 quantities (a large size of memory). Table 1 summarizes the factors and levels used in the experiment.

**Table 1.** Experimental Factors and Levels

| Factor                  |      | Level / Unit                        |
|-------------------------|------|-------------------------------------|
| Content Adaptation      | Low  | 1 (request/min)                     |
| Request Rate            | High | 3 (request/min)                     |
| Number of Adaptation    | Low  | 2 (sub-process)                     |
| Sub-process             | High | 5 (sub-process)                     |
| Process Time of Adapta- | Low  | 25 (seconds/adaptation sub-process) |
| tion Sub-process        | High | 53 (seconds/adaptation sub-process) |
| Network Latency Time    | Low  | 0.74 (seconds)                      |
|                         | High | 7.4 (seconds)                       |
| Size of Buffer          | Low  | 10 (quantity)                       |
|                         | High | 30 (quantity)                       |

In this study we focus on evaluating the performance of three content adaptation approaches for a total of 32 different scenarios. We designed those scenarios using combination of five factors and each factor's corresponding two levels.

### 3.3 Simulation Processes

In this section, each simulation process and its applied formula(s) will be discussed. First, formula 1 was applied to both server-side and proxy-based approaches for calculating the mean time that server needs to execute content adaptation. The formula is defined as following: where  $\bar{T}$  is the total time of a server implementing full content adaptation sub-routines;  $\mu$  is  $1 /$  (processing time of sub-routines adaptation), and  $M$  is the number of adaptation sub-routines.

$$\bar{T} = \frac{M}{\mu} \quad (1)$$

A full content adaptation process can be divided into many sub-routines, so it is reasonable to use this formula to represent the mean time that server needs to perform content adaptation. As opposite to server-side and proxy-based approach, the formula mentioned above is not suitable and appropriate for service-based approach. The basic concept of service-based approach is to perform a specific optimization algorithm to compute the best adaptation path and assign each of the content adaptation sub-routines to dedicated service providers which are distributed in the network. So, content adaptation of service-based approach is accomplished in distributed environment. For simulation, it is very difficult to model a distributed environment. But, under the fundamental assumptions, every server providing adaptation service is assumed to be a M/M/1 queuing system and each one queue is a FIFO queue. All of service providers in the network can be regarded as one open queuing network (Jackson, 1957).



Then, we applied Little's formula (Little, 1961) to the open queuing network to calculate the mean time in the network. The formula for calculating the mean time of content adaptation in the open network is defined as formula 2:

$$\overline{W} = \frac{\overline{N}}{\lambda} = \frac{1}{\lambda} \sum_{i=1}^M \frac{\lambda_i}{\mu_i - \lambda_i}, \quad (2)$$

In formula 2,  $\overline{W}$  is the mean time of one single content remains in the open network;  $\overline{N}$  is the mean number of contents in the open network. And  $\lambda_i$  is the content adaptation request rate for each service provider in the open network;  $\mu_i$  is  $1 /$  (process time of adaptation sub-routines of each service provider in the open network);  $M$  is the number of adaptation sub-routines, and  $i = 1, 2, \dots, M$ .

We encountered two problems over here. First, we do not know the value of every  $\lambda_i$  for each service provider. But, according to Burke's Theorem (Burke, 1957), we could assume that the value of every  $\lambda_i$  is the same. Because any single content never re-enters a previously visited service provider. In other words, how any single content will enter which service provider is determined by the optimized adaptation path algorithm. Therefore, Poisson process is applicable here because it is based on Burke's theorem. Secondly, we do not know the value of every  $\mu_i$ , either. Actually, there is no way to know the value of every  $\mu_i$ . So, we also assume that the processing time of adaptation sub-process of all service providers in the open network are the same. Now, we have simplified Little's formula to be formula 2:

$$\overline{W} = \frac{M}{\mu - \lambda} \quad (3)$$

Though we adopted the simplified Little's formula to help us calculate the mean time in the network. But, it is not good enough, the transmitting time within the network has not been considered yet. We still have to rewrite Little's formula to adapt to the experiment. We added  $d$ , the transmitting time, into the formula. The rewritten formula is finally evolved as formula 4:

$$\overline{W} = M \left( \frac{1}{\mu - \lambda} + d \right), \text{ where } d \text{ is network latency time} \quad (4)$$

Finally, the simulating duration of the simulation has to be designed as well. The chosen simulation running duration was subject to two constraints. First, the test duration must be long enough to accurately assess the server's ability to support the target request rate. Secondly, the duration of each test should be as short as possible, in order to test a total of 32 scenarios, and accurately identify the performance of each content adaptation approach. Therefore, the duration is set to be 24 hours with 100 iterations for each scenario.

## 4 Results and Analysis

For performance measurement, we employed three different measures, 1) total time, 2) network waiting time, 3) number of lost requests, to assess the performance of three content adaptation approaches. Please note that among these three measurements, the results are almost identical for both server-side and proxy-based approaches. Therefore, it is difficult to distinguish the minute differences between both

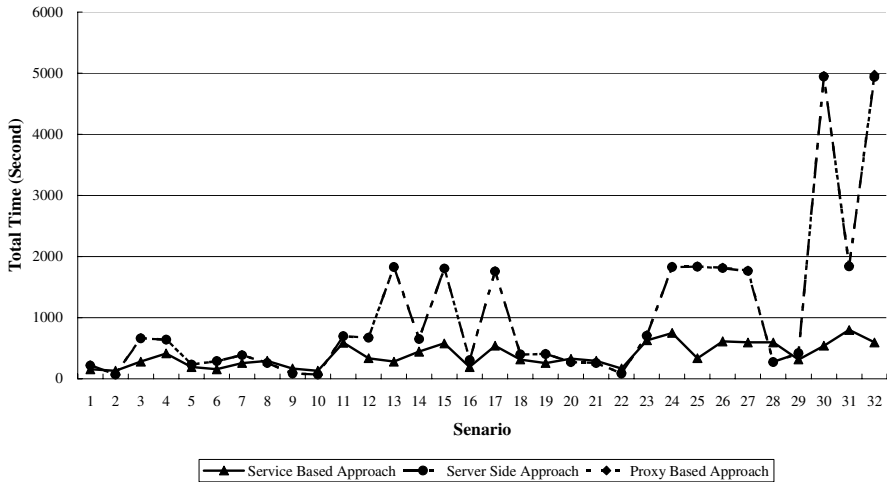


Fig. 2. The Total Time of Implementing Content Adaptation

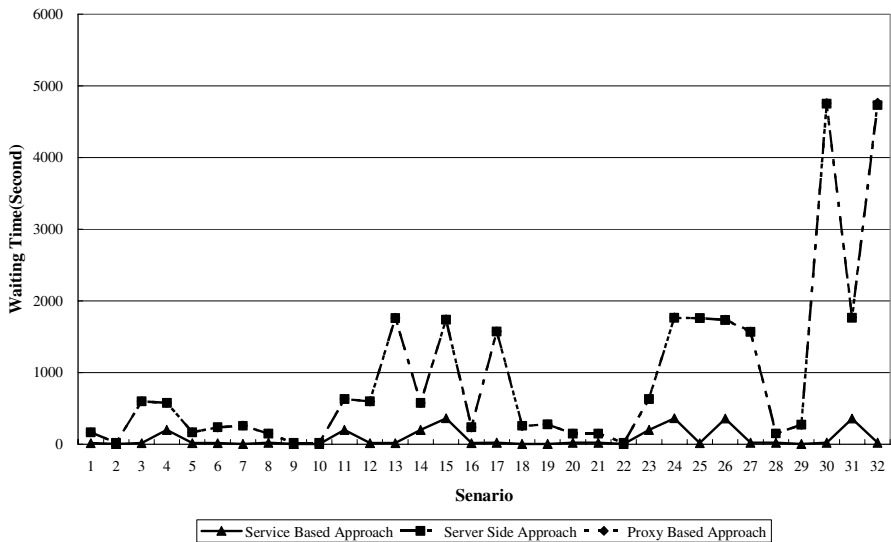


Fig. 3. The Waiting Time of Implementing Content Adaptation

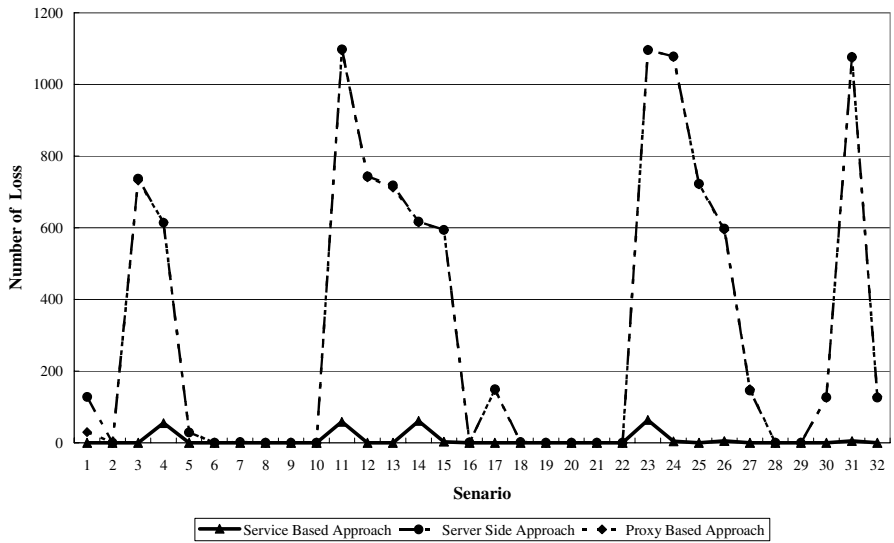


Fig. 4. The Lost of Content Adaptation Requests

lines. Figure 2 shows the total time of implementing content adaptation indicates that a client needs to spend how much time to get content adaptation response. Furthermore, it is the most important measurement to assess the performance of those three approaches. If waited too long, the user is expected to give up waiting for the response at all. Figure 3 shows the waiting time of implementing content adaptation which is the second measurement. This measurement is to tell how much time a content adaptation request is used for waiting but not for executing content adaptation within the network.

Figure 4 shows the third and last measurement, the loss of content adaptation request. Because every server only has finite memory to cache requests, once the received requests exceed the finite limitation of memory the cache could possibly hold, the cache will start to dump previous requests. Adding this measurement is to assess which approach is more powerful and efficient. This is another important measurement that would affect the user satisfaction.

As a result of the basic conception of server side and proxy side approach are the same; the only difference is the location of implementing content adaptation, so the data coming out in simulation does not have great difference more. Both of server side and proxy based approach all get terrible performance in most scenarios. Although, under some scenarios (especially, content adaptation request rate is low and number of adaptation sub-process is high), the total time and waiting time of two approaches seem to have good performance. But, from figure 4, we will find that the number of loss of content adaptation is very high in those scenarios. As opposed to above-mentioned two approaches, service based approach has better performance in three kinds of measures. From figure 2, 3 and 4, we found that the measures of service based approach are all lower than the other two approaches in almost every scenario. In addition, we also found that the performance of service based approach is not easy

to be affected by varying the value of every factor. These finding shows that service based approach have capability to adapt varied network environments and have outperformed the other two approaches.

## 5 Conclusion

As far as we know, this might be the first study to evaluate content adaptation approaches using factorial design simulation. With the nature of complexity for network optimization, this study is far from comprehensive. However, with the power of simulation we believe that it still provides an insightful look at the performance and effects of tradeoffs among three different content adaptation approaches and their influential factors because these approaches are compared fairly under the same settings. The result indicates that service based approach has achieved the best performance, though with the advance of technology, it is not impossible that another new approach won't be proposed or configured in the near future. Nevertheless, this study is a ground work for the optimization of content adaptation network. Moreover, the modeling of adaptation path is also helpful for people who want to redo a similar simulation. Future work should be focused on integrating optimized content adaptation algorithms into the simulation model which is expected to be more realistic and accurate.

## References

1. Acharya, S., "Application and Infrastructure Challenges in Pervasive Computing", NSF Workshop on Context-Aware Mobile Database Management (CAMM), Providence, Rhode Island, January 24-25, (2002).
2. Buchholz, S., Schill, A., "Web Caching in a Pervasive Computing World", Proc. of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SC12002), Orlando, FL, USA, Jul 14-18, (2002.)
3. Bickmore, T., Girgensohn, A., Sullivan, J.W.: Web Page Filtering and Re-Authoring for Mobile Users. In: The Computer Journal 42 (6) (1999)
4. Cohen, A., Rangarajan, S., Singh, N.: Supporting Transparent Caching with Standard Proxy Caches. In: Proceedings of the 4th International Web Caching Workshop, San Diego (1999).
5. Candan, K., Subrahmanian, V., Rangan, P.: Collaborative Multimedia Systems: Synthesis of Media Objects. In: IEEE Transactions on Knowledge and Data Engineering 10 (3) (1998).
6. Duska, B., Marwood, D., Freeley, M.J.: The Measured Access Characteristics of World Wide-Web Client Proxy Caches. In: Proceedings of the USENIX Symposium on Internet Technologies and Systems. Monterey, CA (1997).
7. Elson, J., Cerpa, A., "ICAP - the Internet Content Adaptation Protocol", Internet Draft draft, the ICAP Protocol Group, June (2001). (URL: <http://www.i-cap.org/spec/icap specification.txt>).
8. Fox A., Gribble, S.D., Chawathe, Y., Brewer, E.A.: Adapting to Network and Client Variation Using Active Proxies: Lessons and Perspectives. In: A Special Issue of IEEE Personal Communications on Adaptation (1998)

9. Gribble, S.D., Welsh, M., von Behren, R., Brewer, E.A., Culler, D., Borisov, N., Czerwinski, S., Gummadi, R., Hill, J., Joseph, A., Katz, R.H., Mao, Z.M., Ross, S., Zhao, B.: The Ninja architecture for robust Internet-scale systems and services. In: *Computer Networks* 35 (4) (2001)
10. Geobel, S., Buchholz, S., Ziegert, T., Schill, A.: Device Independent Representation of Web-based Dialogs and Contents. In: *Proceedings of the IEEE Youth Forum in Computer Science and Engineering (YFORIC'01)*, Valencia, Spain (2001)
11. Held, A., Buchholz, S., Schill, A., "Modeling of Context Information for Pervasive Computing Applications", *Proc. of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SC12002)*, Orlando, FL, USA, Jul 14-18, (2002)
12. Han, R., Bhagwat, P., LaMaire, R., Mummert, T., Perret, V., and Rubas, J., "Dynamic Adaptation in an Image Transcoding Proxy for Mobile WWW Browsing". *IEEE Personal Communication*, 5(6):8--17, (1998).
13. Lum, W.Y., and Lau, F.C.M., "A Context-Aware Decision Engine for Content Adaptation", *IEEE Pervasive Computing*, Vol. 1, No. 3, July-September (2002), 41-49.
14. Jain, R. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation and Modeling*, John Wiley & Sons, Inc., New York, NY, (1991).
15. Mohan, R., Smith, J., Li, C.S., "Adapting Multimedia Internet Content For Universal Access", *IEEE Transactions on Multimedia*, March (1999), pp. 104-114.
16. Noble, B.D., Price, M., and Satyanarayanan, M., "A Programming Interface for Application-aware Adaptation in Mobile Computing", *Proc. 2nd USENIX Symposium on Mobile and Location-Independent Computing*, Ann Arbor, MI, USA, Apr. (1995).
17. Ooi, W.; van Renesse, R.: Distributing Media Transformation Over Multiple Media Gateways. In: *Proc. Of the 9th ACM International Multimedia Conference*, Ottawa, Canada (2001).
18. Rahman, R., Menon, R.R. and Rafalow, L., "Enabling OPES to Use Web Service for Call-out Service", draft-rrahman-webservice-00.txt (work in progress), May (2002)
19. (<http://standards.nortelnetworks.com/opes/non-wg-doc/draft-rrahman-web-service-00.txt>).
20. Schilit, B.N., Trevor, J., Hilbert, D.M., Koh, T.K.: m-Links: An Infrastructure for Very Small Internet Devices. In: *Proc. Of the 7th Annual Int'l Conference on Mobile Computing and Networking*, Rome, Italy (2001)
21. Sato, Y.: DeleGate Server. (1994). URL: [http://www.delegate.org/y.sato/DeleGate/\[2002-10-7\]](http://www.delegate.org/y.sato/DeleGate/[2002-10-7])
22. William, L. Oellermann, Jr. "Architecting Web Services", Springer-Verlag New York, Inc., 175 Fifth Avenue, New York, NY, (2001).

# Second Order Snapshot-Log Relations: Supporting Multi-directional Database Replication Using Asynchronous Snapshot Replication

Yochai Ben-Chaim and Avigdor Gal

Technion – Israel Institute of Technology  
{yochai@siglab, avigal@ie}.technion.ac.il

**Abstract.** Multi-directional asynchronous replication is a desired mechanism which allows updates to be performed locally at any site, and later propagated asynchronously to other sites. This paper proposes using second order snapshot-log relations as a mechanism for extending the use of single-directional asynchronous replication to multi-directional. The proposed mechanism is fully based on DBMS core tools and existing DBMS snapshot replication support, thus allowing a natural extension for systems that already support asynchronous snapshot replication. We have implemented and tested the proposed mechanism, showing results and terms of correctness.

## 1 Introduction

Distributed databases store data in multiple locations, allowing prompt response to users' needs. To increase retrieval speed and data availability, distributed databases may maintain *replicas*, copies of data, in several locations simultaneously. In the presence of replicas, user queries are computed using the replica that is *closest* to the user, in terms of either geographical proximity or network distance.

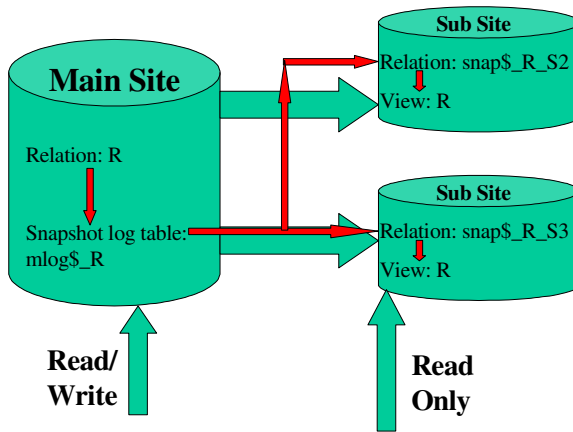
The database research community has put a great deal of effort into developing asynchronous protocols, in which replica updates are performed independently. The advantage of using asynchronous updates is that the transaction can be completed quickly and effectively at the local site, without needing to involve the whole system at once. The trouble with this concept, of course, is that if sites are updated independently, some replicas will contain stale data. Therefore, to be fully effective, asynchronous protocols must build ways to ensure timely updates at all sites (so that user queries are not answered with stale data) and to recover from replica inconsistencies due to local update failures.

Currently, the most common solution to the replica update problem is a process known as *snapshot view replication*. This process works as follows. The distributed database has a single main site to which updates are allowed. A site can serve as a main site for one sub-group of replicated relations, and a sub-site for others. In most cases, a single site serves as the main site for all replicated relations. Each relation maintains a snapshot-log, which holds all changes made to the relation. Replicas are read-only views, updated only by the DBMS using the snapshot-logs of the main site.

Figure 1 illustrates the snapshot view replication as implemented within the Oracle system [3]. The snapshot-log relation  $mlog\$R$  is maintained by the DBMS, as it is used by the DBMS internal mechanisms to create the snapshot of  $R$ . At pre-defined periods (unrelated to the number and/or timing of changes), the DBMS in each sub-site (these are also known as *slaves*) uses the list of changes in the snapshot-log relation to update the replicated relation.

Using snapshot view replication simplifies the update process considerably, yet it may become too restrictive. For one thing, updates can be performed at a single site only, which serves as a single point of failure. Redundant servers provide only a partial solution, and often a costly one, as these can require a great deal of maintenance. The one site that is available for updates must be accessible from all other sites, both so these sites can be updated with replicated data, and so users on these sites can update data. Moreover, the load on this machine is likely to be high, putting stress on both the machine and the network connections.

An alternative to snapshot view replication would allow asynchronous multi-directional updates. In this case, updates of replicated relations would be allowed at any site, and replica updates would be performed asynchronously. As in the uni-directional model, security could be assured through techniques for controlling user permissions for updating relations.



**Fig. 1.** Snapshot view replication in Oracle

Multi-directional replica updates have been previously suggested within synchronous protocols. Such updates are performed using the standard 2PC (2 Phase Commit) protocol [2], which is subject to deadlocks and network failures. Other synchronous protocols in the literature (*e.g.*, [19], [12]) attempt to improve on the 2PC model. Asynchronous bi-directional and multi-directional replica updates have also been suggested [4], [6]. However, these solutions have been based on tools such as vectors, objects and components, grouping of sites, and occasionally *special* reconciliation methods—none of them standard database components within the DBMS. Furthermore,

these suggestions have never been implemented, and so have never undergone full performance analysis or comparison testing.

Vendor databases also offer multi-directional update replication (Oracle's update anywhere and SQL Server's merge replication). At this time, these extensions are criticized as costly (*e.g.*, Oracle's update anywhere is available only in the Enterprise edition which costs \$20K more than the standard edition<sup>1</sup>), hard to design,<sup>2</sup> and bottleneck to performance (*e.g.*, Oracle's benchmark shows that update anywhere replication reduced a 200 update transactions per second machine down to 14 tps<sup>3</sup>).

This paper aims at providing a simple mechanism that can support multi-directional asynchronous database replication using asynchronous snapshot replication mechanism and standard database tools. The use of generic database tools — a feature we share with many others in database research — allows the DBMS to optimally manipulate the database resources and provides an encapsulated unified solution across platforms. The mechanism, dubbed *second order snapshot-log relations* is described in details in Section 3.

To highlight the capabilities of the proposed mechanism, we focus on the problem of preserving replica auto counter primary key attributes, as follows. An auto counter primary-key attribute is defined as a primary-key index attribute for the relation. Activating a triggered process on inserted tuples ensures the correct setting of values. Auto counter primary-key attributes are commonly accepted in DBMSs as offering the most reliable and efficient access to data in a relation ([15], [18]). The problem in correctly maintaining the auto counter attribute in an asynchronously replicated system is that changes to tuples can be performed at a site without knowledge of or regard to possible changes that have been made at other sites. For instance, a tuple may be added without the user being aware that at a different site a tuple has already been added and received the same auto counter attribute value. The solution we propose for this problem, using second order snapshot-log relations, is described in Section 4.

We advocate the use of the proposed mechanism as an easy-to-implement solution for cases where a full-fledged multi-directional replication mechanism is either too expensive or simply an over-kill for the application at hand. Using this mechanism, database designers can maintain their existing support of snapshot replication and enhance it to multi-directional replication mechanism when and where one is needed.

The remainder of this paper is organized as follows. Section 2 describes related work and lays the background for the proposed model. In sections 3 and 4, we present the suggested system architecture and update algorithm, and describe how asynchronous replication can be implemented within this architecture. We continue by providing the model's properties and complexity analysis in Section 5. Section 6 presents the simulation model and the result of our performance evaluation. We conclude in Section 7.

---

<sup>1</sup> <http://www.dbasupport.com/oracle/ora9i/ors.shtml>

<sup>2</sup> [http://asktom.oracle.com/pls/ask/f?p=4950:8:::::F4950\\_P8\\_DISPLAYID:14672061404704](http://asktom.oracle.com/pls/ask/f?p=4950:8:::::F4950_P8_DISPLAYID:14672061404704)

<sup>3</sup> <http://www.dbmsmag.com/9707d01.html>



## 2 Related Work

Synchronized protocols consume many network and time resources, and are subject to deadlocks and extended locking periods while awaiting confirmation from all participants. Asynchronous models allow more rapid updates at one site at the cost of compromising information accuracy at other sites. Updates are performed at sites other than the original site at a later time, with the goal of mutual consistency. Taking into account both data currency and accessibility, the asynchronous model is still considered better than the synchronous model [4], [5], despite the risk of stale data at times, because of the high availability it offers.

The epidemic approach [8] has been suggested for asynchronous data management in distributed databases. According to this method, user operations are executed on a single replica, while in parallel, periodic pair-wise comparison of replicas is performed to detect and update obsolete copies. The epidemic approach requires increased overhead due to replica comparison, which is linear in the number of data items in the database. An alternative to this approach was suggested in [16], where the protocol detects whether update propagation between two replicas is needed at any given time, independent of the number of data items in the database. Should update propagation be needed, it is performed within a period of time linear in the number of data items to be copied. As a result, the number of data items that are frequently updated (and so need to be copied during update propagation) is much smaller than the total number of items in the database. However, the protocol is also based on asynchronous updates from a single updateable replica.

The Bengal database replication system [9] offers a stable mechanism for replication in wireless networks. As opposed to their work, we rely on database core functionality to exchange updates. Our novelty lies in the local architecture that allows a scalable and affordable transformation from uni-directional to multi-directional updates.

The mechanism proposed in this paper improves upon previously proposed bi-directional and multi-directional models [4] in two main respects. First, it is based on components and protocols that are already implemented in a vendor DBMS. Second, the proposed approach is progressive and non-blocking. By progressive we mean that the transaction's coordinator always commits, sometimes with a group of other sites. Asynchronously, the update is later propagated to all other sites. By non-blocking we mean that each site can take unilateral decisions at each step of the algorithm. A reconciliation mechanism is responsible for bringing sites that cannot commit certain updates to mutual consistency, using history logs that are stored locally at each site. It is worth noting that if local replicas enforce the same set of integrity constraints, no reconciliation will be needed.

Standard software development methodologies, especially those that deal with information systems, stress the importance of using DBMS core tools as part of a strategy for dividing system implementation into separate and independent components ([14]). Using standard DBMS core tools allows full independent operation of the database, making the system scalable and enabling the replacement or upgrade of individual components. Among the advantages of using DBMS core tools, we can list ([7], [10]): improved flexibility of the system; reduced cost of system development and maintenance; ease of standards enforcement (including data naming,

structure and formatting conventions); improved security and usage of security levels; and, finally, improved integrity, especially data integrity. The growing field of information engineering ([14]) also favors using the core features of the DBMS for maximizing system capabilities, as well as for efficient resource management.

In this paper we provide, as an example, the use of the proposed mechanism to maintain the uniqueness of auto counter primary keys. While solutions that involve designated offsets for various replicas [20] are common and may solve the uniqueness problem, it does not maintain the correct ordering of insertions to the database. Therefore, such solutions may not work whenever the application requires the tracking of the order of insertions through the primary key.

### 3 Second Order Snapshot-Log Relations

Consider the *base system* of asynchronous replication updates, as described earlier in Figure 1. A single primary site allows updates for each relation, which is later propagated to and updated at all other sub-sites. Figure 2 illustrates the proposed multi-directional extension of the uni-directional model case. In the multi-directional model, updates are available at all sites, and are later propagated to all other sites (relations  $snap\$R\_Si$  where  $i \in \{1, 2 \dots n\}$ ). In this model there is no distinction between a master site and slave sites. All sites allow updates, and all receive propagated updates from all other sites. To achieve this, the relation  $R$  is defined as a read-write relation at all sites.

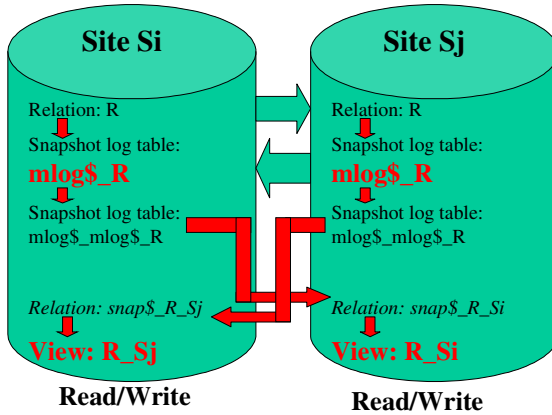


Fig. 2. Multi-Directional Asynchronous Replication Model

We next describe our solution to enhance existing asynchronous snapshot replication mechanism to handle multi-directional updates. As a first step, we create a snapshot-log relation  $mlog\$R$  (dubbed *1<sup>st</sup> order snapshot-log relation*) holding all changes made to  $R$ . This snapshot-log relation is the same one used in uni-directional snapshot-based asynchronous replication. Next, we create an additional snapshot-log relation (dubbed *2<sup>nd</sup> order snapshot-log relation*), based on the  $mlog\$R$  relation and

holding all changes made to  $mlog\$R$  (and indirectly to  $R$ ). The 2<sup>nd</sup> order snapshot-log relation is used to replicate the  $mlog\$R$  relation to other sites (denoted  $mlog\$_mlog\$R$  in Figure 2), as shown by the arrows in Figure 2.

In the third step, we create a snapshot of all instances of  $mlog\$R$  relations from any site to all other sites (denoted  $snap\$R_{Sj}$  and  $snap\$R_{Si}$  in Figure 2). That is, the relation  $snap\$R_{Sj}$  in site  $Si$  (hence also the view  $R_{Sj}$ ) is (periodically) identical to the snapshot log relation  $mlog\$R$  on site  $Sj$ , and vice versa.

The necessity of using a 2<sup>nd</sup> order snapshot-log relation  $mlog\$_mlog\$R$  is illustrated in Figure 1, stems from the need to define the relation  $R$  to be of type read/write at all sites. Recall that in the uni-directional model, updates to the base relation are only available at a single site, while at all other sites the relation is read-only. Therefore, the system alone has the authorization to update  $R$  using  $mlog\$R$ . This is no longer the case for the multi-directional model. Here,  $R$  can be freely updated by the user and the system replica maintenance can no longer guarantee consistency. Therefore, to ensure consistency, each site should have (1) an automatic mechanism to receive updates from all other sites, and (2) a mechanism to propagate these updates to  $R$ . The former is achieved by using  $R_{Si}$  as the read-only relation, capturing updates to  $R$  in site  $Si$ . The latter is achieved by adding a trigger-based algorithm that propagates these updates to local  $R$  relations.

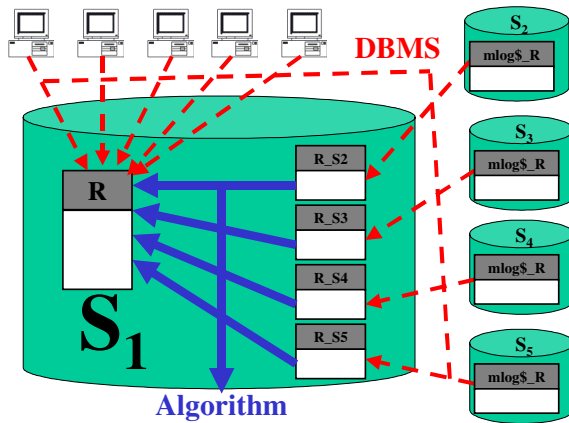


Fig. 3. Algorithm extension of DBMS core standard features

To recap, Figure 3 illustrates an example of the extension over the DBMS core standard features, involving five sites. DBMS core tools allow users to perform read/write operations on relation  $R$  in site  $S_1$ . The DBMS core tools also automatically create and update the snapshot log relations named  $mlog\$R$  in sites  $S_2$ ,  $S_3$ ,  $S_4$ , and  $S_5$ , and create a **read-only** view of those relations (marked  $R_{S2}$ ,  $R_{S3}$ ,  $R_{S4}$  and  $R_{S5}$  respectively) at site  $S_1$ . The algorithm provides the tools for correctly updating the base relation  $R$  from the list of changes received from the other sites.

Using this architecture, each site  $S_i$ , of the  $n$  sites  $S_1, S_2 \dots S_n$ , holds  $n-1$  snapshots ( $snap\$R_{Si}$ ) of each of the  $mlog\$R$  relations from the other  $n-1$  sites. In accordance with the snapshot replication mechanism, only the changes in the 2<sup>nd</sup> order

snapshot-log relation are actually replicated from a given site to each other site. Each site's DBMS, upon receiving the changes in the 2<sup>nd</sup> order snapshot-log relation, uses these changes to construct the correct view of the snapshot-log relation. This occurs automatically by means of the snapshot replication mechanism.

An important feature of the 2<sup>nd</sup> order snapshot-log relation is its ability to allow each site to construct a list of the changes according to a timestamp. This allows the site to employ deterministic logic to consolidate changes made at the local site ([11]) with those made at other sites, sequentially applying the changes in the local relation. We assume that all sites have a common datetime clock, based on known synchronization algorithms (*e.g.*, [13]). The common datetime clock ensures a chronological update ordering. It is worth noting that the common datetime clock does not mean that the updates between the different sites need to be synchronized. On the contrary, the algorithm is independent of the order in which sites receive updates from other sites, and the algorithm does not enforce synchronization of the update process.

## 4 Update Propagation

We term the process by which updates from a server are propagated to other servers *update propagation*. To describe the update propagation process, let us consider the case of a single change made on relation  $R$  at site  $S_i$  (denoted  $S_i.R$ ), and let us show how this change is propagated for execution on each of the other local relations  $S_j.R$  for each  $j \neq i$ . We start with an overview of the process (Section 4.1), followed by an example (Section 4.2). The complete description of the process is given in [1].

### 4.1 Process Overview

As a change made to relation  $S_i.R$  is committed, the DBMS snapshot replication mechanism creates a tuple in the local  $mlog\$_R$  (snapshot-log) relation at site  $S_i$  containing the information on the change made. A tuple created in  $mlog\$_R$  reflects a change to  $S_i.R$ . This operation is accompanied by the creation, by the DBMS, of a new tuple to the relation  $mlog\$_mlog\$_R$ , containing information on changes to  $mlog\$_R$ . The information in these tuples includes the type of change, the values of each attribute before and after the change (where applicable), and the timestamp of the change. A tuple representing a change of type *insert* contains all the inserted tuple's attribute values. A tuple representing a change of type *delete* contains only the auto counter attribute value. A tuple representing a change of type *update* contains the auto counter attribute value as well as the attribute values of the updated attributes, both from before and after the change. It should be noted that snapshot-log relations (both 1<sup>st</sup> and 2<sup>nd</sup> order) are append-only relations, as all change types in the base relation generate new tuples in the snapshot-log relation.

The DBMS common uni-directional snapshot replication model uses the  $mlog\$_mlog\$_R$  relation to replicate the  $mlog\$_R$  relation from  $S_i$  to all other sites  $S_j$  such that  $j \neq i$ . This method ensures that the change in  $mlog\$_R$  relation from  $S_i$  is propagated to all of the other  $n-1$  sites, among them site  $S_j$ .

As site  $S_j$  receives a change in the  $mlog\$R$  replicated relation from  $S_i$ , a triggered operation is run, verifying whether this change has already been executed on  $S_j.R$ . This verification process is conducted according to the type of change and the change's unique timestamp. It compares the local copy of changes to  $R$  ( $mlog\$R$ ) and the local relation holding the list of changes on  $R$  from the remote site  $S_i$  –  $snap\$R\_Si$ . To speed up the retrieval process from these relations, indices are added to the snapshot-log relation and the replicated relations on the timestamp of the change.

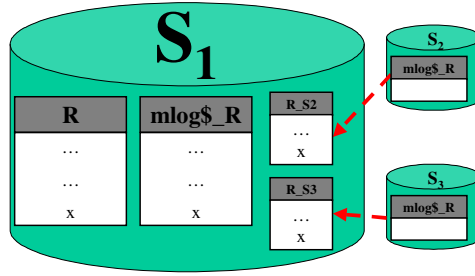
Should the verification process yields that a change has already been executed on  $S_j.R$ , then site  $S_j$  disregards it. Otherwise, site  $S_j$  performs the change on the local base relation, following the sequence of timestamps received from site  $S_j$  to maintain the order among changes. The timestamp serves for later propagating this change to all other sites, as well as for inserting the tuple with the correctly calculated auto counter attribute value. The auto counter attribute value is calculated to reflect the relative datetime value of the inserted tuple. Therefore, if additional tuples are inserted after this tuple, its auto counter attribute value will be altered to reflect the addition. Changes imposed on the auto counter attribute values of tuples should not be propagated to other sites, as the triggered operations at those sites will make the same adjustments upon receiving the change in relation  $R$ . According to the algorithm, the order in which the site receives the updates performed at other sites is immaterial (see Section 5 for algorithm properties).

The triggered operation is executed as an atomic transaction at the local site, ensuring that all updates from a given site are processed before those from a different site. Tuples from any given site are handled in the order of their timestamp. Handling the changes according to the timestamp assures that updates or deletions of tuples received from other sites are related one-to-one to tuples that have already been added in the local site. The local site can only receive changes of type *update* or *delete* that are one-to-one related to tuples for which the local site has already received a change of type *insert*. However, it is possible for an update to be received at a site where the relevant tuple has been deleted. In this case, it may appear that the later update implicitly suggests re-insertion of a tuple accompanied by an update. However, the proposed algorithm assumes that deletion of tuples has a higher priority over later updates to the same tuples.

## 4.2 Triggered Operation Algorithm Example

The algorithm is triggered upon the arrival of new tuples to the replicated copies of snapshot-log relations. The triggered operation is executed as an atomic transaction at the local site. The full algorithm contains a lot of bookkeeping and is provided in [1]. Similar approaches are available in the literature using other data structures (e.g., version vectors [17]). This section provides an example run, describing how sites are updated.

As an example, consider a network environment consisting of three sites:  $S_1$ ,  $S_2$ , and  $S_3$ . We assume that at time  $t$ , all three sites are synchronized. We consider a single relation  $R$ , having at timestamp  $t$  a maximum auto-counter attribute value of  $x$ .



**Fig. 4.** Relations at site  $S_1$

Figure 4 illustrates the relations at site  $S_1$ . The site contains the base relation  $R$ , the snapshot log relation  $mlog\$_R$ , and the replicated views  $R\_S2$  and  $R\_S3$ , which are copies of the snapshot log relations from sites  $S_2$  and  $S_3$  respectively.

Accordingly, site  $S_2$  contains the base relation  $R$ , the snapshot log relation  $mlog\$_R$ , and the replicated views  $R\_S1$  and  $R\_S3$ , and site  $S_3$  contains the relations  $R$ ,  $mlog\$_R$ ,  $R\_S1$  and  $R\_S2$ .

We consider the following list of changes executed at timestamps  $t < t_1 < \dots < t_6$ :

$t_1$  - a tuple is inserted at site  $S_1$  and receives an auto counter attribute value of  $x+1$ .

$t_2$  - the tuple with an auto counter attribute value of  $x+1$  is updated at site  $S_1$ .

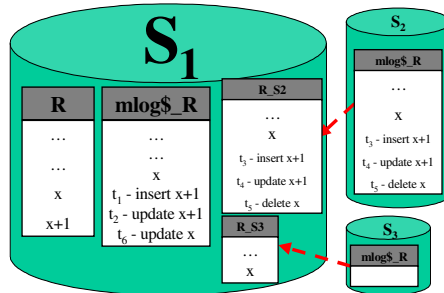
$t_3$  - a tuple is inserted at site  $S_2$  and receives an auto counter attribute value of  $x+1$  (without knowledge that a different tuple has already received this value at site  $S_1$ ).

$t_4$  - the tuple with an auto counter attribute value of  $x+1$  is updated at site  $S_2$ .

$t_5$  - the tuple with an auto counter attribute value of  $x$  is deleted at site  $S_2$ .

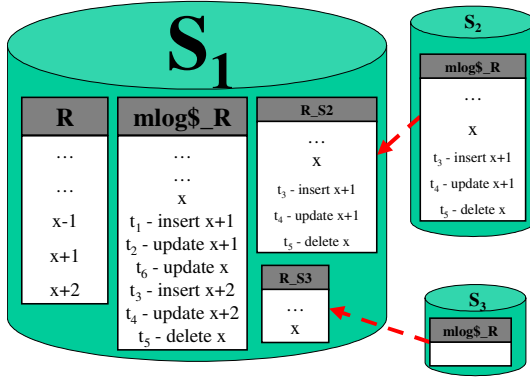
$t_6$  - the tuple with an auto counter attribute value of  $x$  is updated at site  $S_1$ .

Next, the list of changes  $mlog\$_R$  from site  $S_2$  is replicated to site  $S_1$ .



**Fig. 5.** After snapshot log relations are replicated

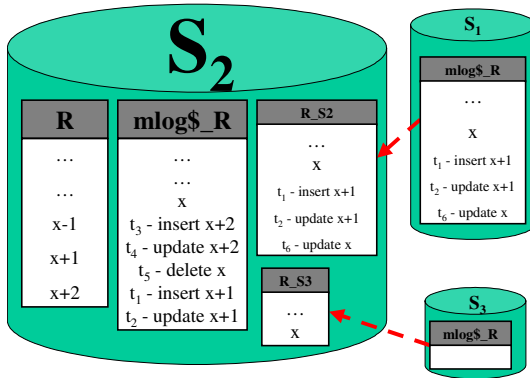
Figure 5 illustrates the relations at site  $S_1$  after the snapshot log relation  $mlog\$_R$  is replicated from site  $S_2$  to site  $S_1$ . A triggered operation at site  $S_1$  on new tuples in relation  $R\_S2$  activates the algorithm. The algorithm checks the new tuples and calculates their net effect on the local base relation  $R$ .



**Fig. 6.** Site  $S_1$  at the end of the triggered operation

The algorithm checks the list of changes from  $R\_S2$  against the list of local changes from relation  $mlog\_R$ . According to the order of timestamps, the algorithm concludes that the tuple added at  $t_3$  should receive an auto counter attribute value of  $x+2$  and not  $x+1$ . It then concludes that the tuple updated at  $t_4$  is the tuple with the new auto counter attribute value of  $x+2$ . Next, the algorithm deletes the tuple with an auto counter attribute value of  $x$ . Figure 6 illustrates site  $S_1$  at the end of the triggered operation.

As site  $S_2$  receives the replicated tuples in relation  $R\_S1$ , the triggered operation concludes that the auto counter attribute value of the local tuple with an auto counter attribute value of  $x+1$  is updated to  $x+2$ , the new tuple that arrived from site  $S_1$  is inserted (and then updated) with an auto counter attribute value of  $x+1$ , and the update of the tuple with an auto counter attribute value of  $x$  is disregarded, as this tuple has been deleted.



**Fig. 7.** Site  $S_2$  at the end of the triggered operation

Figure 7 illustrates site  $S_2$  at the end of the triggered operation on inserted tuples from relation  $R\_S1$ . It is worth noting that while the list of activities in both sites is

not identical ( $S_1$  contains the update in  $t_6$  while  $S_2$  does not) the net effect is the same, since the tuple with the key  $x$  has been deleted.

Site  $S_3$  receives the changes either from site  $S_1$  and then from site  $S_2$ , or vice versa. If the former, the actions at site  $S_3$  will be similar to those that occurred at site  $S_1$ . If the changes from site  $S_2$  are received first, the actions at site  $S_3$  will be similar to those that occurred at site  $S_2$ . At the end of the triggered operations, and in the absence of other update operations, all three sites will contain the same tuples in the base relation  $R$ .

## 5 Main Properties of the Model

The main properties of the model (formally proven in [1]) are:

- **Soundness** - Let  $R$  be a relation with local copies  $S_1.R, S_2.R \dots S_n.R$ . Assume that a change  $u$ , identified by a change type  $t_u$  and timestamp value  $dt_u$ , was performed on relation  $S_i.R$ ,  $1 \leq i \leq n$ . There exists a site  $S_j$  (possibly  $S_j = S_i$ ) at which  $u$  is a user-triggered change.
- **Completeness** - Consider a relation  $R$  with local copies  $S_1.R, S_2.R \dots S_n.R$ . Assume that a change  $u$ , identified by a change type  $t_u$  and timestamp value  $dt_u$ , was initiated by a user on relation  $S_i.R$ ,  $1 \leq i \leq n$ . For each  $S_j$  ( $j \neq i$ ,  $1 \leq j \leq n$ ) there exists a timestamp  $t$  in which  $u \in S_j.snap\$R$ .
- **Termination** - Assume a timestamp  $t_1$  after which users initiate no further changes. There exists a timestamp  $t_2$  ( $t_2 > t_1$ ) such that for each  $t_3 \geq t_2$ ,  $snap\$R_{t_3} = snap\$R_{t_2}$ .  
We define two tuples  $t_1$  and  $t_2$  in relation  $R$  on two different sites  $S_i$  and  $S_j$  as *identical* ( $S_i.R.t_1 = S_j.R.t_2$ ) if the following two conditions hold: a) the two tuples have the same creation timestamp; and b) the two tuples have the same values for all attributes, with the possible exception of the auto counter key attribute.  
We define two relations  $S_i.R$  and  $S_j.R$  to be *consistent* if for every tuple  $t_1$  in  $S_i.R$  there exists an identical tuple  $t_2$  in  $S_j.R$  such that  $S_i.R.t_1.acfv = S_j.R.t_2.acfv$  (where *acfv* is the auto counter value) and vice versa.
- **Correctness** - Assume that changes at all sites have stopped at timestamp  $t_1$ . There exists a timestamp  $t_2$  at which  $S_i.R$  and  $S_j.R$  are consistent.

The complexity analysis of the update propagation mechanism describes the cost in terms of storage space and number of triggered operations. In terms of storage space: assume  $n$  sites, and assume that  $m$  changes were made on the local relation  $S_i.R$  and that the relation  $S_i.R$  holds  $k$  tuples. The uni-directional snapshot replication model requires a total of  $n \times k + m$  tuples at the  $n$  sites. Assuming  $O(m) = O(k)$ , we arrive at a cost linear both in  $n$  and in  $m$ .

In the multi-directional model, the storage space needed sums to  $n \times k + n \times m$  tuples. Again, assuming  $O(m) = O(k)$ , we arrive at a cost linear in  $n$  and in  $m$ . Clearly, the multi-directional model requires additional  $(n-1) \times m$  more tuples than the uni-directional model.

In terms of number of triggered operations: in the uni-directional model each triggered operation performs a single change at each of the  $n$  sites, or  $O(m)$  for  $m$  update operations.



In the multi-directional model, each change results in a triggered operation at each of the  $(n-1)$  sites. Each change is then propagated again to each of the other  $n-1$  sites, thus resulting in  $O(mn)$  triggered operations for  $m$  update operations. The second time a change arrives at a site, it is identified as a change that has already been performed at the site, and is not performed again. According to the algorithm, each triggered operation may affect additional tuples. In the worst case, each triggered operation affects  $O(n)$  tuples, and we arrive at a total of  $O(mn^2)$ , which has a similar theoretical performance to that of the benchmark results of Oracle's update anywhere. Optimization to this process, such as the use of partitions according to the timestamp values to speed up searches, is beyond the scope of this paper.

## 6 Simulation Model

We have implemented the proposed mechanism and tested it using a simulation with three different sites and one base relation replicated at all sites. Each site had a Windows 2000 Server machine, single CPU, with Oracle Server version 8.0.5 that was configured to allow replication operations. The proposed multi-directional model was implemented using standard DBMS tools (Oracle DBMS and PL SQL tools).

The network we used is a standard TCP-IP Microsoft Windows network, and a LAN type network configuration. The database connection definitions were based on TCP-IP addresses.

The simulation shows the relative performance of the multi-directional protocol vis-à-vis the uni-directional model in scalability and practical functional values of the algorithm. We measured:

- CPU usage of the servers.
- The amount of data transferred over the network.
- The average wait-time for up-to-date data in the nearest server when performing a query.

### 6.1 Experiment Setup

Each run began with an empty relation at each site. This is not a prerequisite of the algorithm, yet served to simplify the calculations. During the test runs, the relations at each site were populated. All the changes were of type *insert* to simplify the simulation. We expect to achieve similar performance with a combination of update operations as well.

In what follows, we use the following terminology:

*Network Usage*: the total number of bytes transferred.

*Reconciliation*: the process by which each server sends updates performed on the local relation to all other servers, and performs updates received from all other servers. The reconciliation process ends when no updates remain to be propagated.

*Low Stress*: server operates below full capacity.

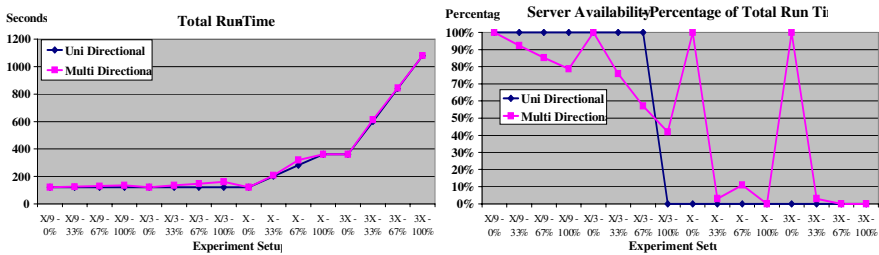
*High Stress*: server is required to perform more operations per second than it can handle.

*Database Server Availability*: A server is considered *available* when its CPU usage falls below 100% utilization. Requests to a server whose CPU is being fully utilized

are queued for later handling. A server's availability is measured in the amount and percentage of time in which the server's CPU is utilized at less than 100%.

## 6.2 Experiment Results and Performance Analysis

The CPU usage experiments show that for low-stress situations— $X/9$  and  $X/3$ , where  $X$  is the number of requests per second that brings the server to 100% utilization (approximately 32 in the experiment setup used) — the multi- and uni- directional models exhibit almost identical availability. For high-stress situations,  $X$  and  $3X$ , there are significant differences, and the multi-directional protocol allows for periods of time in which the server CPUs are utilized at less than 100%. This is because in the multi-directional model the load is divided among several servers, whereas in the uni-directional model a single master server performs all updates.



**Fig. 8.** Server availability and overall run time until the reconciliation process ended for different stress levels (relative to  $X$ ) and degrees to which update requests make up of the total requests

Figure 8 illustrates our results for the master server in the uni-directional model, and a representative server for the multi-directional model. The figure shows server availability as a percentage of run time (on the right), and overall run time (on the left), for 16 experiments at different stress levels. As can be seen, the servers in the multi-directional model show better availability under high stress conditions, while the uni-directional model shows slightly shorter run times (despite the order of magnitude theoretical difference stated in Section 5).

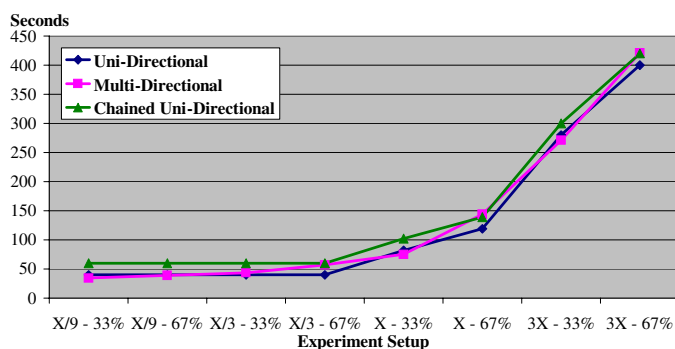
## 6.3 Network Topology

The experiments were performed over a LAN network where all servers and all the users were connected over the same LAN, so that network traffic was not such that would cause bottlenecks or high network stress. Under these conditions, our experimental results show that the network traffic imposed by the multi-directional model is roughly  $n$  times (where  $n$  is the number of sites) greater than that needed for the uni-directional model.

The experiments for this paper showed that network usage is not a factor over a LAN, as the server usage limit is exceeded far before the network bandwidth is exhausted. An additional parameter over networks is the delay time. Over a LAN, the delay time is negligible.

Taking into account both network usage and delay time, the multi-directional model shows an incremental advantage over the uni-directional model in the transaction and propagation processing procedures. In the uni-directional model, each transaction is completed only after being committed on the master server. In the multi-directional model, on the other hand, each transaction initiated by a user is usually completed within the local sub-network, and so is not subject to delay times or network bottlenecks.

Using the results of the experiments performed over a LAN network, we were able to calculate the average wait-time for up-to-date data in the nearest server for each user. Looking again at Figure 8, recall that we used 16 different experimental setups, with four stress levels ( $X/9$ ,  $X/3$ ,  $X$ , and  $3X$ ) and, within each stress level, four degrees to which update requests make up of the total requests (0%, 33%, 67%, and 100%). Cases where updates made up all requests or none are irrelevant for this experiment and so can be discarded, leaving us with 8 experimental setups.



**Fig. 9.** Wait-Time for Up-To-Date Data Comparison between the Multi- and Uni- Directional Models over a WAN network

Figure 9 shows the average wait-time (in seconds) for the most up-to-date data under uni- and multi-directional protocols for these 8 setups. Over a LAN network, the uni-directional model shows slightly shorter wait-times than does the multi-directional model.

Figure 9 represents the wait-time for up-to-date data comparison between the multi- and uni- directional models, as expected over a WAN environment with an average delay time of five seconds. As can be seen, roughly half of the experiment setups show a better average wait-time for the multi-directional model over the uni-directional model. For the chain variation, the uni-directional model shows longer wait-times than the multi-directional model in almost all the experiments. In general, the multi-directional model will show better wait-times than the uni-directional model as the delay over a WAN network increases.

## 7 Conclusion

In this paper we propose a multi-directional extension to the uni-directional asynchronous replication model, synchronizing updates in the presence of auto counter primary

keys. We have introduced second order snapshot log relations as the mechanism and presented an algorithm to maintain multi-directional asynchronous replication, while conserving the relations' auto counter attribute correct value (for use as a primary-key attribute). The model makes it possible to build a distributed system that is updated asynchronously (update margin intervals are scalable to system needs) without having to compromise on the use of auto counter primary key attributes or on the ability to alter each relation at each site, at a cost measured only in disk space and minor DBMS resources. An important attribute of this mechanism is that it is based fully on DBMS core tools, thus allowing a natural extension for systems that already support the uni-directional asynchronous update model.

We have implemented and experimented with the proposed mechanism in a simulated environment. Experimental results show the terms under which better performance and server availability is expected from the multi-directional model. Additionally, over a WAN network, the uni-directional model experiences a much slower update process as compared to the multi-directional model. In the multi-directional model, the delay times are only evident in the asynchronous update process. User-initiated update transactions are usually performed over a LAN. In the uni-directional model, the user suffers slower response times when performing update transactions, as these usually need to be committed over a WAN.

## References

- [1] Removed for the sake of double-blind review process.
- [2] Bernstein, P. A., Hadzilacos, V. and Goodman N. (1987), "Concurrency Control and Recovery in Database Systems", Addison-Wesley.
- [3] Bobrowski, S. and Smith G. (Primary Authors - 1997), "Oracle8 Replication, Release 8.0, Part No. A58245-01", Oracle Corporation.
- [4] Ceri, S., Houtsma, M. A. W., Keller, A. M. and Samarati, P. (1992), "Achieving Incremental Consistency among Autonomous Replicated Databases", *Proceedings of the IFIP WG 2.6 Database Semantics Conference on Interoperable Database Systems (DS-5)*, pp. 223-237.
- [5] Ceri, S., Houtsma, M. A. W., Keller, A. M. and Samarati, P. (1995), "Independent Updates and Incremental Agreement in Replicated Databases", *Distributed and Parallel Databases*, 3(3), pp. 225-246.
- [6] Chang, T. P. and Hull, R. (1995), "Using Witness Generators to Support Bi-directional Update Between Object-Based Databases", *Proceedings of the fourteenth Symposium on Principles of Database Systems (PODS)*, pp. 196-207.
- [7] Dadam, P. (2000), "On the Design, Implementation, and Maintenance of Enterprise-wide Transactional Workflow Applications for Advanced Environments: Challenges and Open Issues", position paper, <http://www-adele.imag.fr/IPTW/IPTW/Papers/>.
- [8] Demers, A., Greene, D., Hauser, C., Irish, W., Larson, J., Shenker, S., Sturgis, H., Swinerhart, D. and Terry, D. (1987), "Epidemic Algorithms for Replicated Database Maintenance", *Proceedings of the 6<sup>th</sup> Symposium on Principles of Distributed Computing (PODS)*, pp. 1-12.
- [9] Ekenstam, T., Matheny, C., Reihner, P., and Popek, G.J. (2001), "The Bengal Database Replication System", *Distributed and Parallel Databases*, 9(3), pp. 187-210.
- [10] Elmasri, R. and Navathe, S. (2000), "Fundamentals of Database Systems (3rd Edition)", Addison-Wesley.

- [11] Goldring, R. (1995), "Things Every Update Replication Customer Should Know", *Proceedings of the International Conference on Management of Data (SIGMOD)*, pp. 439-440.
- [12] Hsu, M. and Silberschatz, A. (1991), "Unilateral Commit: A New Paradigm for Reliable Distributed Transaction Processing", *Proceedings of the Seventh International Conference on Data Engineering (ICDE)*, pp. 286-293.
- [13] Lamport, L. (1990), "Concurrent Reading and Writing of Clocks", *ACM Trans. On Computer Systems*, vol. 8, pp. 305-310.
- [14] Martin, J. (1989), "Information Engineering: Introduction", Prentice Hall Professional Technical Reference.
- [15] Melonfire, I. (2002), "PHP Application Development With ADODB", Developer Shed Network Site, [www.devshed.com](http://www.devshed.com).
- [16] Rabinovich, M., Gehani, N. H., Kononov, A. (1996), "Scalable Update Propagation in Epidemic Replicated Databases", *Proceedings of the 5<sup>th</sup> International Conference on Extending Database Technology: Advances in Database Technology (EDBT)*, pp. 207-222.
- [17] Ratner, D., Reiher, P., and Popek, G. (1997), "Dynamic Version Vector Maintenance", Computer Science Department: University of California, Los Angeles, 1997.
- [18] Reed, J. (2003), "Carbon User Manager Rdbms Usage", Sapient.
- [19] Singhal, M. (1990), "Update Transport: A New Technique for Update Synchronization in Replicated Database Systems", *IEEE Transactions on Software Engineering (TSE)*, 16(12), pp. 1325-1336.
- [20] Soparkar, N. and Silberschatz, A. (1990), "Data-value Partitioning and Virtual Messages," *Proceedings of 9th ACM SIGA CT-SIGMOD-SIGART Symposium on Principles of Database Systems*, Nashville, TN.

# The Replica Management for Wide-Area Distributed Computing Environments<sup>\*</sup>

Jaechun No<sup>1</sup>, Chang Won Park<sup>2</sup>, and Sung Soon Park<sup>3</sup>

<sup>1</sup> Dept. of Computer Software  
College of Electronics and Information Engineering  
Sejong University, Seoul, Korea

<sup>2</sup> Intelligent IT System Research Center  
Korea Electronics Technology Institute  
Bundang-gu, Seongnam-si, Korea

<sup>3</sup> Dept. of Computer Science & Engineering  
College of Science and Engineering  
Anyang University, Anyang, Korea

**Abstract.** Data replication is a critical issue in distributed computing where a large amount of data sets are frequently shared among geographically distributed scientists. The usual way of managing consistent data replicas between distributed sites is to periodically update the remotely located data replicas. However, this method doesn't guarantee the replica consistency in such a case that remote clients who retain the replicas irregularly update or modify their replicas. In this paper, we introduce two kinds of data replication techniques, called owner-initiated data replication and client-initiated data replication, that are developed to support data replica consistency without requiring any special file system-level locking functions. Also, we present performance results on Linux clusters located at Sejong University.

## 1 Introduction

Providing a consistent data sharing mechanism is a critical issue in distributed computing where a large amount of data sets generated from large-scale data-intensive scientific experiments and simulations are frequently shared among geographically distributed scientists [1, 2, 3]. In such an environment, the data replication technique significantly affects the performance by minimizing the remote data access time.

The usual way of managing consistent data replicas between distributed sites is to periodically update the remotely located data replicas, like implemented in Globus toolkit [4, 5, 6, 7, 8]. This method is implemented under the assumption that the data being replicated is read-only so that once it has been generated would not it be modified in any grid site. This assumption is no longer true in such a case that a remote site may modify or update the data replicated and

---

<sup>\*</sup> This work was supported in part by a Seoul R&BD program.

stored in its local storage. If another remote site tries to use the same data replicated on its storage before the data is updated with the new one, the data consistency between distributed sites would then be failed and the scientist gets the wrong results.

In order to solve this problem by providing the data replication consistency, we have developed two kinds of data replication techniques, called owner-initiated data replication and client-initiated data replication, and integrated those techniques with GEDAS (Grid Environment-based Data Management System) [9, 10] that is a grid toolkit providing a high-level, user-friendly interface to share the remotely produced data sets among the grid communities.

In the owner-initiated data replication, the data owner who owns the application data sets starts the data replication to share the data sets with remote clients. In the client-initiated data replication, the client who needs the data sets starts the data replication by connecting to the data owner. Furthermore, our data replication techniques do not need to use file system-level locking functions so that they can easily be ported to any file systems.

The rest of this paper is organized as follows. In Section 2, we discuss the previous GEDAS metadata structure. In Section 3, we present the design and implementation of our replication techniques integrated with GEDAS. Performance results on the Linux cluster located at Sejong University are presented in Section 4. We conclude in Section 5.

## 2 Previous Work

GEDAS maintains the centralized metadata structure that is stored at the data owner location where the application data sets are generated. When an application first generates the data sets, GEDAS registers the application to the `application_registry_table` with the application name. Each time an application writes data sets, GEDAS enters the modification date, dimension, problem size, number of timesteps, and IP address of the owner location to the `run_registry_table`.

The `data_registry_table` includes the properties of each data set, such as logical dataset name, data type, storage order, data access pattern, and global size. The `file_registry_table` stores a globally determined file offset denoting the starting offset of the file of each data set. GEDAS uses this information to make appropriate MPI-IO calls to access the real data. The `file_registry_table` also includes the physical file name to be mapped to the data set.

In the previous version, GEDAS used the version checking to achieve the replica consistency that is much similar to the locking mechanism of distributed file systems [11, 12]. Whenever a remote client modifies the replicated data, it informs the data modification to all remote sites that share the same data, using the version number. This method however incurs significant communication overheads because all remote sites including the sites who are not interested in the data set modified must receive the version number.

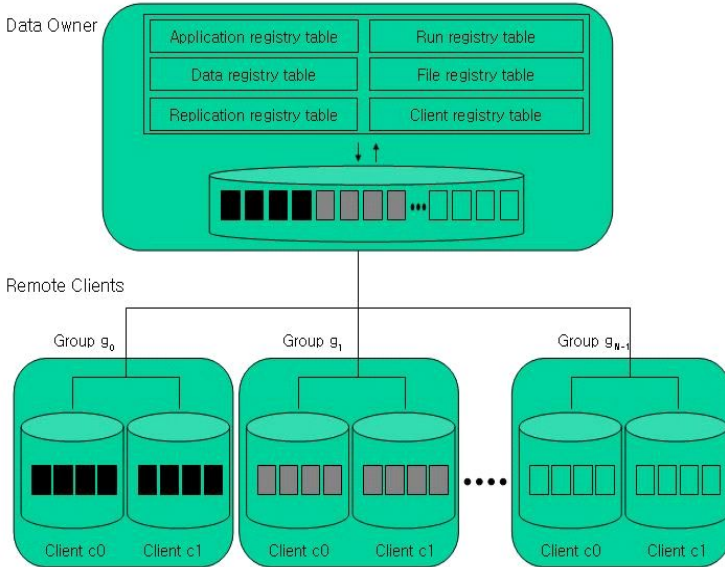
### 3 Data Replication

#### 3.1 Client Grouping in GEDAS

The functionality of the current version of GEDAS is enforced to eliminate the unnecessary communication overheads by implementing the client grouping to enable only the remote clients who share the same data sets to receive the data modifications. By making the client groups, GEDAS can easily detect the other clients who share the same data replicas and can let them take the new copy of the modified data, without affecting other clients.

Figure 1 shows the client grouping in GEDAS. In order to support the client grouping and the replication mechanism, the metadata structure of the current GEDAS is extended to six data base tables, like shown in Figure 2.

The metadata database tables and the application data sets generated by users are located at the data owner. The remote clients who want to share the application data sets with the data owner are grouped into several client groups, according to the data sets replicated on the local storage. Figure 1 shows  $n$  remote client groups created based on the data replicas. Each client in a group is identified with groupID and clientID, such as  $(g_0, c_0)$  for the first client and  $(g_0, c_1)$  for the second client in Group  $g_0$ .



**Fig. 1.** Client grouping in GEDAS

#### 3.2 Data Replication in GEDAS

When a client wants to receive the data replicas from the data owner at the first time, he should register to the metadata database table by clicking on the





Fig. 2. GEDAS Metadata Structure

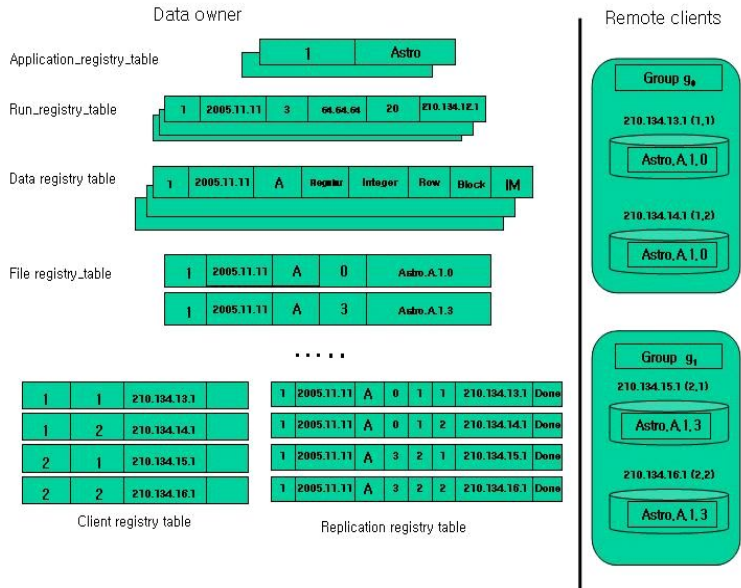


Fig. 3. GEDAS Data Replication

application\_registry\_table, run\_registry\_table, data\_registry\_table, and file\_registry\_table to select the data sets of interest.

If there is a group who has already registered to share the same data sets, the new client will then be a member of the existing group. Otherwise, a new group

where the new client belongs to is created by taking the new group ID in the `client_registry_table`. Also, the `replication_registry_table` is appended to reflect the new client registration. Figure 3 shows the metadata structure stored in the data owner and the data replicas stored in the remote clients.

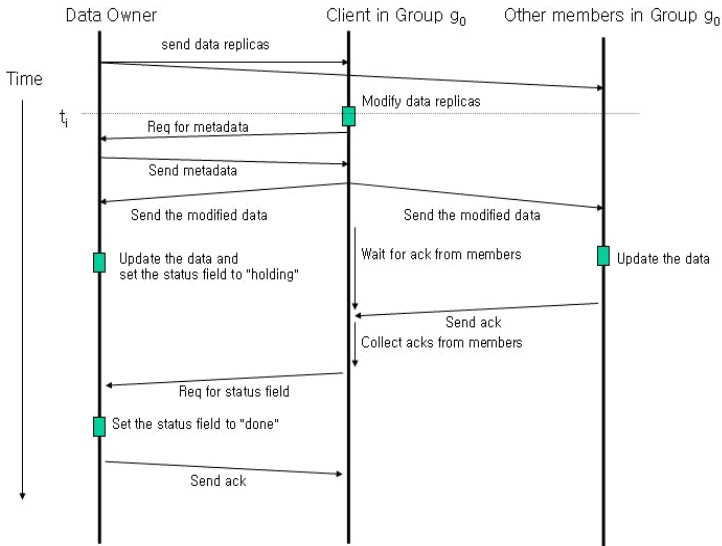
In Figure 3, user runs an astrophysics application (Astro) while producing the data set A at the time steps 0 and 3, respectively. The physical file name storing a data set is automatically created by concatenating the application name, logical data set name, uniqueID, and time step. Therefore, the data set produced at the time step 0 is stored in Astro.A.1.0 and the data set produced at the time step 3 is stored in Astro.A.1.3.

In Figure 3, four remote clients are grouped into two client groups, according to the data replica shared. The two remote clients of the first group, Group  $g_0$ , share the data set generated at the time step 0, Astro.A.1.0, with the data owner, and the other two clients of the second group, Group  $g_1$ , share the data set generated at the time step 3, Astro.A.1.3.

### 3.3 Owner-Initiated Data Replication

In order to maintain the replica consistency among the remote clients, we developed two replication approaches, called owner-initiated data replication and client-initiated data replication.

In the owner-initiated data replication, when user generates data sets on the data owner, GEDAS replicates them to the remote clients who registered to GEDAS to share the data sets with the data owner. When a remote client changes



**Fig. 4.** Owner-Initiated Replication

the data replicas stored in its local storage, it broadcasts the modifications to the members in the same group where it belongs to and to the data owner for replica consistency. Figure 4 shows the steps taken in the owner-initiated replication.

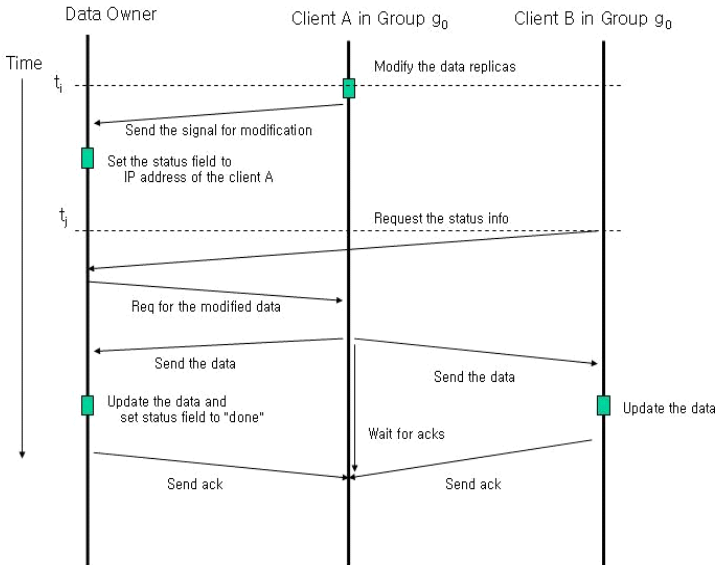
Suppose that a client belonging to Group  $g_0$  modifies the data replicas at time  $t_i$ . The client first sends the request for the IP address of other clients in  $g_0$  to the data owner. After the client receives the IP address, it sends the modified data to the other clients in  $g_0$  and to the data owner and waits for the acknowledgement.

When the data owner receives the modified data, it updates them to the local storage and sets the status field of the replication\_registry\_table to "holding" to prevent another client from accessing the data sets while being updated. When the data owner receives the signal from the client who initiated the data modification, it sets the status field to "done", allowing another client to use the data replica.

The owner-initiated data replication approach allows remote clients to share the data replicas safely, provided that they find out the corresponding status field is set to "done". Moreover, even though a remote client crashes, it doesn't affect to the data consistency since as soon as the data replicas are modified the change is immediately reflected to the data owner and to the other clients in the same group. However, if the data modification to the data replicas frequently happens, the heavy communication bottleneck then incurs even if no one else would use the data sets modified.

### 3.4 Client-Initiated Data Replication

Figure 5 shows the client-initiated data replication where only when the modified data replicas are needed by users are those data replicas sent to the requesting



**Fig. 5.** Client-Initiated Replication

client and to the data owner. Unlike in the owner-initiated data replication, there is no data communication when users on the data owner produce the application data sets. If a client needs to access the remote data sets stored in the data owner, he will then get the data replica while registering to GEDAS.

Suppose that client A belonging to Group  $g_0$  modifies a data replica at time  $t_i$ . He just sends a signal to the data owner to update the corresponding status field of the data set to the IP address of client A.

At time  $t_j$ , suppose that client B accesses the data replica stored in its local storage but not been updated by client A's data modification. In order to check the replica consistency, client B first requests the status information of the data replica to the data owner. The data owner finds out that the data set has been modified by client A and requests the data to client A. Client A sends the modified data to the data owner and to the client B, and then waits for the acknowledgement from both. After the data owner updates the modified data set and sets the status field to "done", it sends back an acknowledgement to the client A.

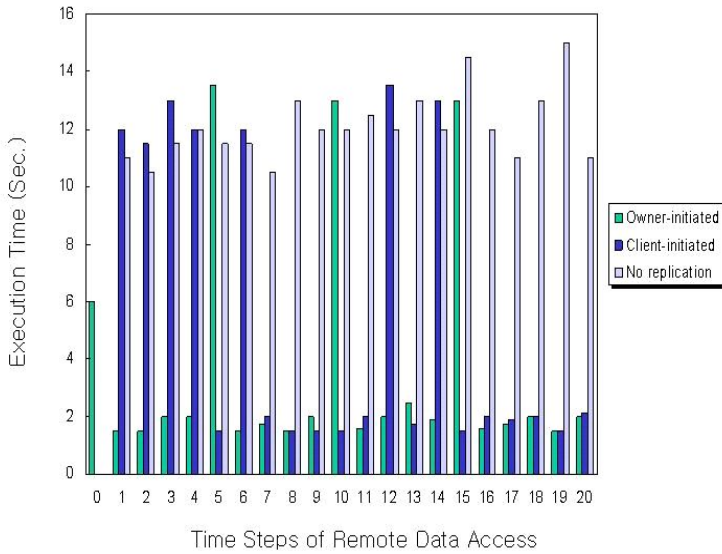
In the client-initiated data replication approach, the data replicas are sent to the remote clients only when the data sets are actually needed by them. Therefore, unlike in the owner-initiated data replication approach, the client-initiated data replication does not incur unnecessary data communication. However, if a client who keeps the modification of the data replica is crashed before the data modification is updated to the data owner and to the other members of the same group, a significant data loss will then be happened.

## 4 Performance Evaluation

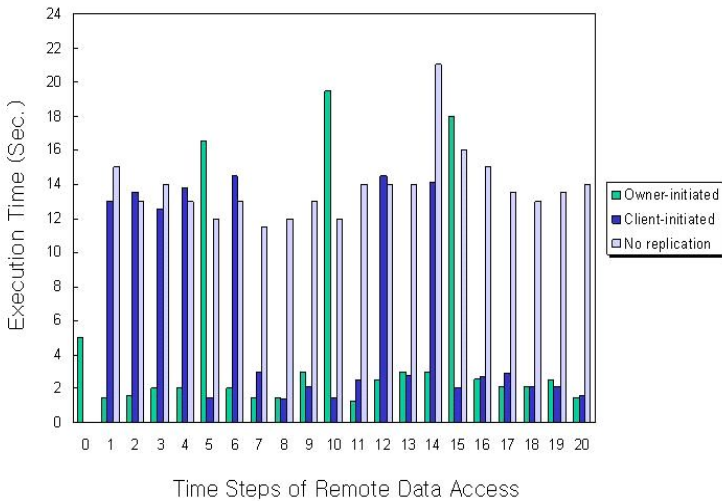
In order to measure the performance, we used two Linux clusters, located at Sejong university. Each cluster consists of eight nodes having Pentium3 866MHz CPU, 256 MB of RAM, and 100Mbps of Fast Ethernet each. The operating system installed on those machines was RedHat 9.0 with Linux kernel 2.4.20-8.

The performance results were obtained using the template implemented based on the three-dimensional astrophysics application, developed at the University of Chicago. The total data size generated was about 520MB and among them, 400MB of data were generated for data analysis and data restart, and then the remaining 120MB of data were generated for data visualization. The data sets produced for data visualization are used by the remote clients, thus requiring to perform data replications to minimize data access time.

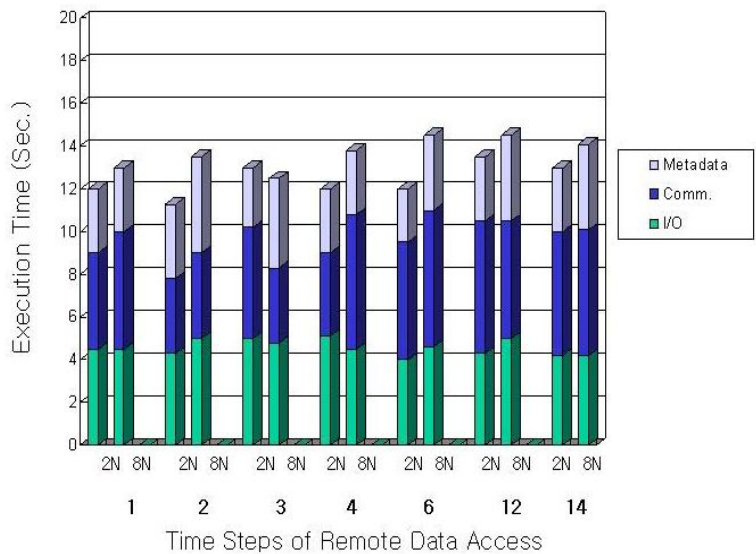
In order to evaluate two replication approaches, a randomly chosen remote client modified 30MB of replicas at time steps 5, 10, and 15, respectively, and spread those replicas to the data owner and to the clients, according to the owner-initiated data replication and to the client-initiated data replication. At each time step, a maximum execution time for the data replication measured among the remote clients was selected as a performance result. This time includes the cost for metadata accesses to the data owner, real data communication, and I/O operations.



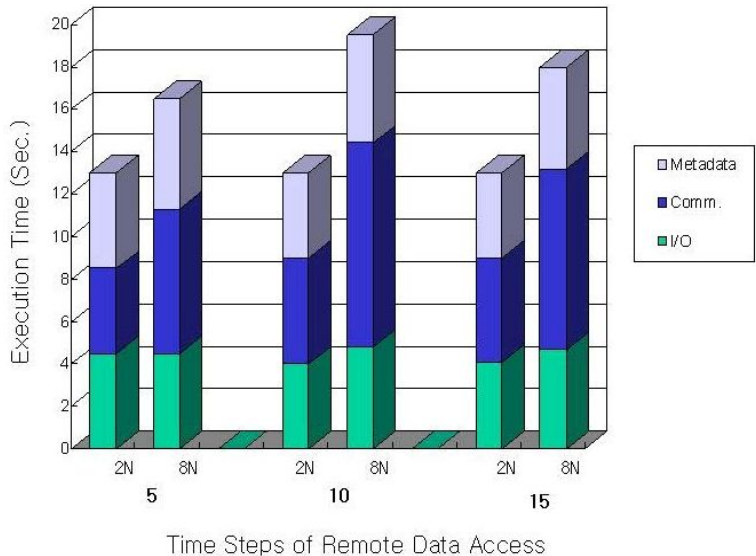
**Fig. 6.** Execution time for replicating visualization data on the remote clients as a function of time steps for accessing remote data sets. Two client groups were made, while each group consisting of two nodes.



**Fig. 7.** Execution time for replicating visualization data on the remote clients as a function of time steps for accessing remote data sets. Two client groups were made, while each group consisting of eight nodes.



**Fig. 8.** Execution time comparison between 2nodes per client group(2N) and 8nodes per client group(8N), using the client-initiated data replication



**Fig. 9.** Execution time comparison between 2nodes per client group(2N) and 8nodes per client group(8N), using the owner-initiated data replication

In Figure 6, we made two client groups, while each group consisting of two nodes. At each time step, a client accesses either 30MB of replicas stored in the local storage, or 30MB of remote data sets stored on the data owner in such a case that the data sets required are not replicated to the local storage.

In the owner-initiated replication, as soon as an application produces data sets at time step 0, all the remote clients receive the necessary visualization data sets to replicate them to the local storage. These replicas are used until the modification to the replicas happens at time steps 5, 10, and 15, respectively.

When the replicas stored in the local storage are used, the execution time for accessing visualization data sets drops to almost about 3 seconds needed for communicating the corresponding data properties, such as file name, data type, and status information to check replica consistency, with the data owner.

If the modification to the replicas happens, like occurred at time steps 5, 10, and 15, respectively, the modified replicas are then broadcast to the data owner and to the clients in the sample group, thereby increasing the execution time for accessing remote data sets.

In the client-initiated replication, since there is no replica stored in the client side until time step 4, each remote client should communicate with the data owner to receive the data sets needed. From time step 5, because each client can use the data replicas stored in its local storage, the execution time for accessing data sets dramatically drops to almost 3 seconds.

When the replicas are modified at time steps 5, 10, and 15, the client-initiated approach just sends to the data owner the IP address of the client modifying the replicas, and thus it takes no more than 3 seconds spent to access metadata. However, we can see that in Figure 6, at time steps 6, 12, and 14, another client tries to access the modified replicas, thus incurring the data communication and I/O costs to update the replicas to the requesting client and to the data owner.

Without data replication, the data communication for accessing the remote data sets consistently happens to the clients, affecting the performance.

In Figure 7, we increased the number of nodes in each group to eight. Using the owner-initiated data replication, we can see that as the number of nodes in each group is increased the execution time for replicating remote data sets also goes up due to the increment in the communication overhead to broadcast the replicas to the data owner and to the other clients.

On the other hand, as can be seen in Figure 7, the client-initiated data replication shows not much difference in the execution time to receive the modified data sets because less number of nodes than in the owner-initiated data replication is involved in the communication.

In Figure 8, we analyzed the execution time of each time step, using the client-initiated data replication. We measured the times spent on the metadata accesses, real data communication, and I/O to update the data modification. We eliminated the time steps when clients accessed the data replicas stored in their local storages, while choosing the time steps involving the real data communication. Like above, we made two client groups and compared the execution times between two nodes per group(2N) and eight nodes per group(8N).

As can be seen in Figure 8, not much difference between two configurations can be found because, with the client-initiated data replication, no matter how many nodes belong to each client group only three nodes including the data

owner, requesting client, and client retaining the data modification are involved in the data communication to update the data replicas.

On the other hand, in the owner-initiated data replication shown in Figure 9, as the number of nodes per group is increased the time for communicating the real data also goes up because the modification to the data replicas is updated to the data owner and to all the clients retaining the same replicas. However, we believe that more performance evaluations should be conducted to present some valuable conclusions with these two replication approaches.

## 5 Conclusion

We have developed two data replication approaches to maintain the replica consistency in case of replica modifications or updates. In the owner-initiated data replication, the replication occurs when applications generate the data sets in the data owner location. Whenever a remote client modifies or updates its replica, in order to maintain the data consistency, it broadcasts the replica to the other members of the same group, as well as to the data owner retaining the entire application data sets. In the client-initiated data replication, only when the data sets are needed by a remote client are the necessary data sets replicated to the requesting client. If a client modifies its replica, then it just sends a signal to the data owner in order to make the other client recognize the recently modified replicas. Due to the data broadcast, the owner-initiated data replication approach shows the increased communication overhead when the number of nodes per group becomes large. However, since the modification to the replicas is immediately reflected to the data owner and to all the clients retaining the same replicas, the owner-initiated data replication can maintain the replica consistency even in the client crashes. On the other hand, the client-initiated data replication shows the constant communication cost even with the increased number of nodes per client group. But in case of client crashes, the client-initiated data replication can lose the recently modified real data. In the future, we plan to use our replication techniques with more applications and evaluate both the usability and performance.

## References

1. B. Allcock, I. Foster, V. Nefedova, A. Chervenak, E. Deelman, C. Kesselman, J. Leigh, A. Sim, A. Shoshani, B. Drach, and D. Williams. High-Performance Remote Access to Climate Simulation Data: A Challenge Problem for Data Grid Technologies. SC2001, November 2001
2. R. Moore, A. Rajasekar. Data and Metadata Collections for Scientific Applications. High Performance Computing and Networking (HPCN 2001), Amsterdam, NL, June 2001
3. A. Chervenak, E. Deelman, C. Kesselman, L. Pearlman, and G. Singh. A Metadata Catalog Service for Data Intensive Applications. GriPhyN technical report, 2002
4. I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration WG, Global Grid Forum, June 22, 2002



5. A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, S. Tuecke. The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets. *Journal of Network and Computer Applications*, 23:187-200, 2001
6. A.Chervenak, R. Schuler, C. Kesselman, S. Koranda, B. Moe. Wide Area Data Replication for Scientific Collaborations. *Proceedings of 6th IEEE/ACM International Workshop on Grid Computing (Grid2005)*, November 2005.
7. M. Cai, A. Chervenak, M. Frank. A Peer-to-Peer Replica Location Service Based on A Distributed Hash Table. *Proceedings of the SC2004 Conference (SC2004)*, November 2004.
8. W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, I. Foster. The Globus Striped GridFTP Framework and Server. *Proceedings of Super Computing 2005 (SC05)*, November 2005.
9. J. No, H. Park. GEDAS: A Data Management System for Data Grid Environments. *Proceedings of International Conference on Computational Science*, 2005, pages 485–492
10. J. No, R. Thakur, and A. Choudhary. High-Performance Scientific Data Management System. *Journal of Parallel and Distributed Computing*, (64)4:434-447, April 2003
11. C. A. Thekkath, T. Mann, and E. K. Lee. Frangipani: A Scalable Distributed File System. In *Proceedings of the Symposium on Operating Systems Principles*, 1997, pages 224–237
12. K. W. Preslan, A. P. Barry, J. E. Brassow, G. M. Erickson, E. Nygaard, C. J. Sabol, S. R. Soltis, D. C. Teigland, and M. T. O’Keefe. A 64-bit Shared Disk File System for Linux. In *Proceedings of Sixteenth IEEE Mass Storage Systems Symposium Seventh NASA Goddard Conference on Mass Storage Systems & Technologies*, March 15-18, 1999

# Developing Parallel Cell-Based Filtering Scheme Under Shared-Nothing Cluster-Based Architecture<sup>\*</sup>

Jae-Woo Chang and Tae-Woong Wang

Dept. of Computer Engineering, Chonbuk National Univ.,  
Chonju, Chonbuk 561-756, South Korea  
jwchang@chonbuk.ac.kr, twwang@dblab.chonbuk.ac.kr

**Abstract.** A large number of high-dimensional index structures suffer from the so called 'dimensional curse' problem, i.e., the retrieval performance becomes increasingly degraded as the dimensionality is increased. To solve this problem, the cell-based filtering (CBF) scheme has been proposed, but it shows a linear decrease in performance as the dimensionality is increased. In this paper, we develop a parallel CBF scheme under an SN(Shared Nothing) cluster-based parallel architecture, so as to cope with the linear decrease in retrieval performance. In addition, we devise data insertion, range query and k-NN query processing algorithms which are suitable for the SN parallel architecture. Finally, we show that our parallel CBF scheme achieves good retrieval performance in proportion to the number of servers in the SN architecture and it outperforms a parallel version of the VA-File when the dimensionality is over 10.

## 1 Introduction

For the image content-based retrieval on XML documents, an object in an image can be defined as an n-dimensional feature vector. In order to acquire the information that best corresponds to one's request by finding some patterns and tendencies among the data, the information used in data warehousing can be composed of n-dimensional attribute vectors having several attributes. Therefore, it is necessary to research on high-dimensional index structures for efficiently retrieving high-dimensional data in such applications as data warehousing and XML document retrieval. For this, high-dimensional index structures have been proposed [1,2,3], but most of them suffered from the so called 'dimensional curse' problem, i.e., the retrieval performance becomes increasingly degraded as the dimensionality is increased [4]. To solve this problem, the Cell-Based Filtering (CBF) scheme and the VA-file scheme were proposed. The VA-file performs filtering by using vector approximation information [5]. The CBF scheme performs filtering by using signatures, and shows good performance by redefining the maximum and minimum distances for good filtering [6]. However, both schemes show a linear decrease in retrieval performance as the dimensionality increases. In order to alleviate this linear decrease in retrieval performance, it is

---

<sup>\*</sup> This work is financially supported by the Ministry of Education and Human Resources Development (MOE), the Ministry of Commerce, Industry and Energy (MOCIE) and the Ministry of Labor (MOLAB) through the fostering project of the Lab of Excellency.

necessary to make use of a parallel processing technique. In this paper, we develop a parallel CBF scheme under an cluster-based parallel architecture, so as to cope with the linear decrease in retrieval performance. For our parallel CBF scheme, we also devise data insertion, range query and k-NN query processing algorithms which are suitable for the parallel architecture.

This paper is organized as follows. In Section 2, we introduce the existing CBF scheme. In Section 3, we describe our parallel CBF scheme developed under an cluster-based parallel architecture. In Section 4, we provide experimental performance results. Finally, we draw our conclusions and suggest future work in Section 5.

2 Cell-Based Filtering (CBF) Scheme

To solve the 'dimensional curse' problem, the VA-file scheme [2] and CBF scheme [3] were proposed. The VA-file minimizes the 'dimensional curse' problem by scanning a sequential file in which an approximation of each cell is stored. The CBF scheme improves the retrieval performance by filtering out cells effectively using the signature of a feature vector, as well as the distance between the feature vector and the center of the cell containing it. Figure 1 shows the overall architecture of the existing CBF scheme.

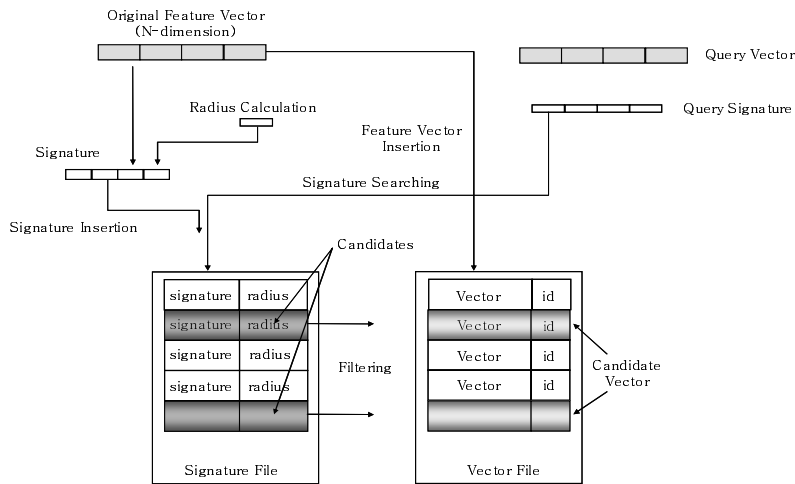


Fig. 1. Overall architecture of CBF Scheme

To transform an object feature vector into a signature, the CBF scheme first divides a high-dimensional data space into cells, and it creates a signature which corresponds to a cell containing the object feature vector. The signature is constructed by concatenating the representatives of each dimension for the cell. When a user query is given, a query signature is generated at first by a signature generation algorithm. Secondly, the CBF scheme performs the filtering process, by sequentially searching each signature stored in a signature file. It regards candidate signatures as those signatures which are

within the search range of the query vector. Finally, it retrieves the real feature vectors corresponding to a given query, by checking the candidate vectors corresponding to candidate signatures.

### 3 Parallel CBF Scheme

#### 3.1 Overall Architecture of Our Parallel CBF

To respond to a user query, the CBF scheme sequentially scans signatures and feature vectors, which are stored in a signature file and a data file, respectively, on a physical disk. However, our parallel CBF scheme distributes signatures and feature vectors over multiple processors by means of a declustering technique, and stores them on separate physical disks. As a declustering technique, either horizontal or vertical partitioning can be employed [7,8]. In order to design an efficient declustering algorithm for our parallel CBF scheme, it is necessary to consider the two properties of the CBF scheme.

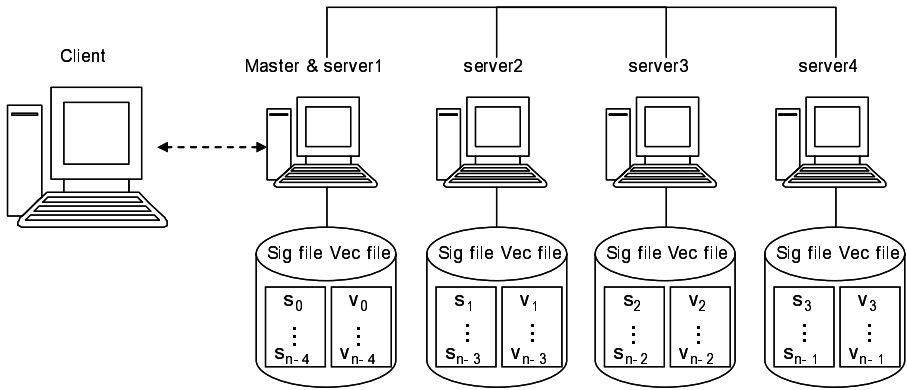
**Property 1: All signatures of the CBF scheme should be scanned sequentially in order to respond to a given query**

When a query is processed, a parallel CBF scheme scans all of the signatures contained in the signature file. Thus, if the signatures are distributed evenly over multiple servers by a horizontal partitioning technique, the number of page accesses required is approximately equal to  $B/S$ , where  $S$  is the number of servers and  $B$  is the total number of blocks containing all of the signatures. If they are distributed into multiple servers by a vertical partitioning technique, however, we need approximately  $B/S$  page accesses, and additional merging time is also required for the partitioned signatures.

**Property 2: The feature vectors corresponding to the candidate signatures in the CBF scheme should be partially searched in order to respond to a given query**

When a query is processed, a parallel CBF scheme scans only those feature vectors corresponding to the candidate signatures contained in the data file. Thus, if the feature vectors are distributed evenly over multiple servers by a horizontal partitioning technique, the number of page accesses required is approximately equal to  $N/S$ , where  $S$  is the number of servers and  $N$  is the number of feature vectors corresponding to the candidate signatures. If the feature vectors are distributed over multiple servers by a vertical partitioning technique, however, we need approximately  $N/S$  page accesses, and additional merging time for the partitioned feature vectors is also required.

Based on the above two properties, we propose a shared-nothing (SN) cluster-based parallel CBF scheme which distributes both signatures and feature vectors over multiple servers by means of a horizontal partitioning technique. Figure 2 shows the overall architecture of our parallel CBF scheme under the SN cluster-based architecture with four processors, i.e., servers. A special server serves as the master, which distributes the inserted data over the multiple servers using a horizontal partitioning



**Fig. 2.** Overall architecture of our parallel CBF

technique. In addition, the master performs the parallel processing of a query received from a client, by first sending the query to multiple servers and allowing them to process it simultaneously. Then, the master computes the final answer based on the results obtained from the servers and transmits the final answer to the client.

### 3.2 Data Insertion

Our parallel CBF scheme has two files in, i.e. the signature file and the data file. To create the signature file, we first generate a signature from a given feature vector by using a signature generation algorithm. Then, the signature generated is merged with the distance between the feature vector and the center of the cell containing it. Finally, the signature, which has been merged with the distance, is stored in the signature file. To store both the signature and data files in our parallel CBF scheme, it is necessary to distribute them over multiple servers under a SN cluster-based parallel architecture. In order to accomplish the uniform distribution of data over multiple servers, we devise an equation to assign an original  $n$ -dimensional vector and its signature into a specific processor by using a horizontal partitioning technique as follows. Here,  $P_i$  means a processor to be assigned,  $N_p$  means the number of processors used for parallelism, and  $Max\_SR$  means the maximum number of shifting and rotating a signature. The  $rand\_num()$  generates a random number ranging from 0 to 1 and makes use of the radius of an  $n$ -dimensional vector as its seed. In addition, '>>' means a bit operator to shift and rotate a binary code, and  $(int)[\ ]$  means an automatic type conversion from a binary code to an integer. Eq.(1) allows for the nearly uniform distribution of a set of vectors in the data file and their signatures over multiple processors because even some  $n$ -dimensional vectors residing on the same cell can be distributed into different processors using the equation.

$$P_i = (int)[Signature_i \gg (rand\_num(\text{the radius of Vector}_i) * Max\_SR)] \% N_p \quad (1)$$

In Figure 2, the master uniformly distributes the inserted data over multiple servers using Eq.(1) and makes the servers to store the data on their own local disks. The master lets the servers operate in parallel, based on both a thread technique and a

socket network communication. A thread is considered as an individual operation in a program and has the same meaning as a process from the conceptual point of view. The difference between them is that a process requires a large amount of data to be stored during context switching, i.e. heavy weighting, but a thread requires only light weighting during context switching. The use of multiple threads allows for performing multiple operations in a program simultaneously. When a feature vector is inserted, the master node creates as many threads as the number of servers. Each thread stores the assigned data into its own buffer. If the buffer is full, each thread transmits the buffer to its corresponding server, which stores the associated information into its signature and data files. The data insertion algorithm in our parallel CBF scheme is as follows. Firstly, we read a feature vector from the data file, determine on which server this vector is to be inserted by means of Eq. (1), and then store the vector into the buffer corresponding to this server situated in the master server. Secondly, if the buffer is full, the buffer is sent to the CBF instance of the corresponding server. Thirdly, we generate a signature for the feature vector and compute the distance between the vector and the center of cell containing it. Finally, we store both the signature and the distance in the signature file of the chosen server and store the feature vector and its identifier (ID) in the data file of the server.

### 3.3 Range Query Search

For answering a range query in our parallel CBF scheme, a user inputs both a query vector and a distance value. The range query searches for all of the objects included within a circle which is centered on the query vector in data space and whose diameter is equal to the distance value. We first transform the query vector into a query signature, and then search the cells within the boundary given by the query signature. To respond to a range query, it is necessary to access multiple servers simultaneously, since both the signature and data files are distributed over multiple servers under a SN cluster-based parallel architecture. When a user query is processed, the master node creates as many threads as there are servers. Each thread sends the query to its own server. Each server searches its own signature and data files in parallel, thus retrieving the feature vectors that match the query. By simultaneously searching multiple sets of signature and data files, our parallel CBF scheme improves the overall retrieval performance because the same amount of data is distributed over multiple servers under a SN parallel architecture. The algorithm of range query processing is as follows. Firstly, we send the range query to the CBF instance of each server. Secondly, each CBF instance creates a signature list from its own signature file. Thirdly, we generate a query signature from the query vector and find candidate signatures by searching the signature list. Fourthly, we obtain a result set by retrieving candidate vectors from the data file which corresponds to the candidate signatures. We also send the result set to the master node. As the last step, we obtain the final result by merging the result sets transmitted from the servers.

### 3.3 k-Nearest Neighbor (k-NN) Search

The k-NN algorithm of our parallel CBF scheme consists of three filtering phases, like the original k-NN algorithm [9], and an additional phase for integrating the result

lists obtained by the three filtering phase. This is because the signature and data files are distributed over multiple servers under an SN parallel architecture. The k-NN query processing algorithm is as follows. In the first phase, we first create a thread for each server and send the k-NN query to each thread. Secondly, the CBF instance of each server generates a signature list by searching all of the signatures in its own signature file. Thirdly, we generate the first candidate signature list by inserting signatures sequentially into the candidate list, without performing any comparison, until the number of signatures to be inserted is k. Once the number of signatures in the candidate list is k, each thread compares the next signature with the k-th candidate signature in the candidate list. If the next signature has a shorter distance from the given query point than the k-th candidate signature, then the k-th candidate signature is deleted from the candidate list and the next signature is inserted into it. Otherwise, we continue to compare the next signature with the k-th candidate signature until there are no more signatures to be compared. In the second phase, we reduce unnecessary page accesses by deleting cells whose lower bound is greater than the k-th upper bound in the candidate signature list. In the third phase, we obtain a result list by retrieving those real vectors which correspond to the candidate signatures of the candidate list. To accomplish this, we first compare the lower bound of the last candidate cell with the k-th object distance. If the k-th object distance is less than the lower bound of the last candidate cell, the last candidate cell is deleted from the candidate list. Otherwise, we calculate the distances between the query point and the real objects and generate a result list by obtaining the nearest k objects. Next, we transmit the result list to the master node. In the case where the number of servers is N, the number of nearest neighbors obtained from the multiple servers is  $k \times N$ . Finally, we integrate the result lists obtained from the multiple servers and find the final k nearest objects in the master node.

4 Performance Analysis

For the performance analysis of our parallel CBF scheme, Table 1 describes our experimental environment. Here, we make use of 10, 20, 50, and 80-dimensional synthetic data sets, each being obtained by randomly generating 2 million points in data space. At first, we compare our parallel CBF scheme with the conventional CBF scheme in terms of the data insertion time and the search times for both range and k-NN queries. Secondly, we compare our parallel CBF scheme with a parallel version

Table 1. Experimental environment

|                    |  |   |
|--------------------|--|---|
| System Environment | 4 servers (each 450 MHz CPU,HDD 30GB,128 MB Memory)<br>Redhat Linux 7.0 (Kernel 2.4.5), gcc 2.96 (g++) |   |
| Data Set           | Synthetic data   | 2 million data<br>( 10, 20, 40, 50, 60, 80, 100-dimensional data) |
|                    | Real data  | 2 million data ( 10, 20, 50, 80-dimensional data)                 |
| Retrieval time     | Range query  | Retrieval time for searching 0.1 % of data                        |
|                    | k-NN query   | Retrieval time for searching 100 nearest objects                  |

**Table 2.** Analytical experimental parameters

| Parameters | Description  |
|------------|--|
| $L_s$      | The length of the signature data (bits)                          |
| $L_v$      | The length of the feature vector data (bits)                     |
| $N$        | The number of data   |
| $b$        | The number of signature bits                                     |
| $d$        | Dimension  |
| $P$        | Page size (8kbyte)   |
| $D$        | The number of disks(4, 8, 10, 16)                                |
| $r$        | Radius from an object in a cell to the central point of the cell |
| $C_i$      | Number of candidate cells on disk $i$                            |
| $C_{\max}$ | Maximum number of candidate cells on disk $i$                    |
| $R_s$      | Number of I/Os with Signature                                    |
| $R_v$      | Number of I/Os with Feature Vector                               |
| $R$        | Number of total I/Os   |

of the VA-File. For this purpose, we implement our parallel CBF scheme under a SN cluster-based architecture with four servers.

#### 4.1 Analytical Performance Metric

We estimate the analytical performance metric for our parallel CBF scheme. This is used to measure the theoretical performance improvement provided by our parallel CBF scheme. The parameters used for the analytical performance estimation are shown in Table 2. Eq. (2) is used to calculate the number of page accesses required to answer a given query, regardless of whether it is a range query or a k-NN query. In order to estimate the analytical performance metric, we represent the total number of page accesses for a query as  $R$ .  $R$  is the sum of  $R_s$  and  $R_v$ , where  $R_s$  is the number of page accesses needed to search a signature file, and  $R_v$  is that needed to search a data file. Here, we assume that the retrieval performance depends solely on the number of page accesses.

$$R = R_s + R_v$$

$$R_s = \left( \frac{L_s * N}{P} \right) / D \quad L_s = (b * d * N) + r \quad (2)$$

$$R_v \approx \max(C_i) \quad (\forall_i, 0 \leq i < D)$$

#### 4.2 Experimental Performance Analysis

To estimate the insertion time, we measure the time needed to insert two millions pieces of synthetic data. Table 3 shows the insertion time for both the existing CBF scheme and our parallel CBF scheme. In the CBF scheme, it takes about 240, 480 and 1860 seconds to insert 10-, 20- and 100-dimensional data, respectively. In our parallel



CBF scheme, the same operations take about 250, 450 and 1930 seconds, respectively. It can be seen from the insertion performance result that our parallel CBF scheme is nearly the same as the existing CBF scheme. This is because our parallel CBF can reduce the time required to insert data by using multiple servers, but it requires additional time due to the communication overhead.

**Table 3.** Insertion time for synthetic data (unit:sec)

| Dimension<br>Scheme | 10     | 20     | 40     | 50      | 60      | 80      | 100     |
|---------------------|--------|--------|--------|---------|---------|---------|---------|
| Existing CBF        | 236.50 | 478.89 | 828.49 | 992.78  | 1170.40 | 1505.90 | 1862.16 |
| Our parallel CBF    | 256.12 | 445.42 | 824.02 | 1001.93 | 1188.42 | 1543.62 | 1930.10 |

The range query is used to search for objects within a certain distance from a given query point. For our experiment, we use a radius value designed to retrieve 0.1% of the data from the two million synthetic data. Table 4 shows the retrieval time for the range query. The performance improvement metric of our parallel CBF scheme against the existing CBF scheme can be calculated by  $1/(PT/CT)*100$ , where PT and CT refer to their performance measurements (retrieval times) of our parallel CBF scheme and that of the existing CBF scheme, respectively. In the existing CBF scheme, it takes about 13, 16, 38 and 60 seconds to respond to a range query in the case of 10-, 20-, 50- and 80-dimensional data, respectively. In our parallel CBF scheme, it takes about 1.6, 2.2, 7.3 and 11.6 seconds for the same operations, respectively. When the number of dimensions is 80, the performance improvement metric of our parallel CBF scheme is about 520%, being greater than the analytic performance improvement metric, i.e. 400%. This is because our parallel CBF scheme utilizes a large buffer under the SN cluster-based architecture with four servers.

**Table 4.** Retrieval time for range query using synthetic data

| Dimension<br>Scheme            | 10    | 20    | 40    | 50    | 60    | 80    | 100   |
|--------------------------------|-------|-------|-------|-------|-------|-------|-------|
| Existing CBF                   | 13.20 | 16.42 | 31.91 | 37.93 | 44.96 | 59.50 | 74.27 |
| Our parallel CBF               | 1.59  | 2.24  | 5.92  | 7.27  | 7.87  | 11.60 | 14.04 |
| Performance improvement metric | 830%  | 733%  | 539%  | 522%  | 571%  | 513%  | 529%  |

The purpose of the k-nearest neighbors (k-NN) query is to search for the k objects which best match a given query. For the purpose of performance analysis, we measure the time needed to respond to a k-NN query for which k is 100. Table 5 describes the retrieval time for the k-NN query using two million synthetic data. In the case of

10-dimensional data, it takes about 3.6 seconds to retrieve the data for the CBF scheme and 1.4 seconds for our parallel CBF scheme. This is because in our parallel CBF scheme, we retrieve objects simultaneously from four servers under the SN cluster-based parallel architecture. In the case of 40-dimensional data, the CBF scheme requires about 31.9 seconds to respond to a k-NN query, while our parallel CBF scheme requires about 8.3 seconds. In the case of 100-dimensional data, the CBF scheme requires about 76.4 seconds to respond to a k-NN query, while our parallel CBF scheme requires about 25.6 seconds. Thus, it is shown that the performance improvement metric of our parallel CBF scheme is about 300-400%, depending on the dimension of the data. The performance improvement for a k-NN query is relatively low, compared with that for a range query. This is because the overall retrieval performance for a k-NN query is very sensitive to the distribution of the data. That is, it entirely depends on the lowest retrieval performance among the four servers. When the number of servers is D, the overall performance of a k-NN query is not linearly increased in proportion to D. This is because an additional time is required to integrate the result lists obtained from the multiple servers.

**Table 5.** Retrieval time for k-NN query using syntactic data

| Dimension<br>Scheme            | 10   | 20    | 40    | 50    | 60    | 80    | 100   |
|--------------------------------|------|-------|-------|-------|-------|-------|-------|
| Existing CBF                   | 3.59 | 17.80 | 31.92 | 39.28 | 46.72 | 61.60 | 76.39 |
| Our parallel CBF               | 1.44 | 2.62  | 8.28  | 10.78 | 11.02 | 18.97 | 25.62 |
| Performance improvement metric | 249% | 679%  | 386%  | 364%  | 424%  | 325%  | 298%  |

4.3 Comparison with Parallel VA-File

In this section, to verify the usefulness of our parallel CBF scheme as a high dimensional indexing scheme, we compare our parallel CBF with the parallel version of VA-file. Table 6 describes the retrieval time for a range query using two million pieces of real data. In the case of the parallel VA-file, it takes about 2.3 seconds to respond to a range query containing 20-dimensional data and about 13 seconds for 80-dimensional data. In the case of our parallel CBF scheme, it takes about 1.8 seconds for 20-dimensional data and 10 seconds for 80-dimensional data. As dimensionality is higher, our parallel CBF scheme achieves better retrieval performance than the parallel VA-file. This is because our parallel CBF scheme performs good filtering by redefining the maximum and minimum distances as the dimensionality is increased.

**Table 6.** Retrieval time for range query using real data

| Dimension<br>Scheme | 10   | 20   | 50   | 80    |
|---------------------|------|------|------|-------|
| Parallel VA-file    | 1.71 | 2.27 | 4.37 | 12.96 |
| Our parallel CBF    | 1.49 | 1.82 | 2.75 | 10.08 |

Table 7 shows the retrieval time for a k-NN query. In the case of the parallel VA-file, it takes about 1.8 seconds to respond to a range query containing 20-dimensional data and about 11 seconds for 80-dimensional data. In the case of our parallel CBF scheme, it takes about 1.8 seconds for 20-dimensional data and 10.5 seconds for 80-dimensional data. Thus, the performance of our parallel CBF scheme is slightly better than that of the parallel VA-file in case dimensionality is high.

**Table 7.** Retrieval time for k-NN query using real data

| Dimension<br>Scheme | 10   | 20   | 50   | 80    |
|---------------------|------|------|------|-------|
| Parallel VA-file    | 1.51 | 1.82 | 7.17 | 11.32 |
| Our parallel CBF    | 1.60 | 1.77 | 6.06 | 10.44 |

## 5 Conclusions and Future Work

For such applications as data warehousing and image content-based retrieval, high-dimensional index structures have been proposed, but most of them suffered from the so called 'dimensional curse' problem, i.e., the retrieval performance becomes increasingly degraded as the dimensionality is increased. To solve the 'dimensional curse' problem, the CBF scheme was proposed. However, as the dimensionality is increased, the retrieval performance of the CBF scheme decreases linearly. To cope with this problem, we proposed our parallel CBF scheme, which could uniformly distribute both signatures and feature vectors over multiple servers using a horizontal partitioning method under the SN cluster-based parallel architecture. We showed from the performance analysis that our parallel CBF scheme provided a near linear improvement in proportion to the number of servers, for both the range and k-NN queries. We also showed that our parallel CBF scheme outperformed the parallel VA-file when the number of dimensions is greater than 10. In the future work, our parallel CBF scheme is needed to be used as a high dimensional indexing scheme in real applications, such as image content-based retrieval on XML documents and data warehousing.

## References

1. D.A. White and R. Jain, "Similarity Indexing : Algorithms and Performance", In Proc. of the SPIE : Storage and Retrieval for Image and Video Databases IV, Vol. 2670, pp.62-75, 1996.
2. H.I. Lin, H. Jagadish, and C. Faloutsos, "The TV-tree : An Index Structure for High Dimensional Data", VLDB Journal, Vol. 3, pp. 517-542, 1995.
3. S. Berchtold, D. A. Keim, H-P. Kriegel, "The X-tree : An Index Structure for High-Dimensional Data, Proceedings of the 22nd VLDB Conference, pp.28-39, 1996.
4. Berchtold S., Bohm C., Keim D., Kriegel H. -P, "A Cost Model for Nearest Neighbor Search in High-Dimensional Data Space", ACM PODS Symposium on Principles of Databases Systems, Tucson, Arizona, 1997.

5. Roger Weber, Hans-Jorg Schek, Stephen Blott, "A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces," Proceedings of 24rd International Conference on Very Large Data Bases, pp.24-27, 1998.
6. S.-G. Han and J.-W. Chang, "A New High-Dimensional Index Structure Using a Cell-based Filtering Technique", In Lecture Notes in Computer Science 1884(Current Issues in Databases and Information Systems), Springer, pp. 79-92, 2000.
7. J.-K. Kim and J.-W. Chang, "Horizontally-divided Signature File on a Parallel Machine Architecture," Journal of Systems Architecture, Vol. 44, No. 9-10, pp. 723-735, June 1998.
8. J.-K. Kim and J.-W. Chang, "Vertically-partitioned Parallel Signature File Method," Journal of Systems Architecture, Vol. 46, No. 8, pp. 655-673, June 2000..
9. N. Roussopoulos, S. Kelley, F. Vincent, "Nearest Neighbor Queries", Proc. ACM Int. Conf. on Management of Data(SIGMOD), pp. 71-79, 1995.

# How Deep Should It Be?

## On the Optimality of Hierarchical Architectures

Amihai Motro<sup>1</sup>, Alessandro D'Atri<sup>2</sup>, and Eli Gafni<sup>3</sup>

<sup>1</sup> Information and Software Engineering Department  
George Mason University, Fairfax, VA 22030, USA  
[ami@gmu.edu](mailto:ami@gmu.edu)

<sup>2</sup> Centro di Ricerca sui Sistemi Informativi  
LUISS “Guido Carli” University, Via Tommasini 1, I-00162 Rome, Italy  
[datri@luiss.it](mailto:datri@luiss.it)

<sup>3</sup> Computer Science Department  
University of California, Los Angeles, CA 90095, USA  
[eli@cs.ucla.edu](mailto:eli@cs.ucla.edu)

**Abstract.** Many areas of information technology implement *hierarchical architectures*. Notable examples are the organization of computer files in folders, the arrangement of program menus, and the distribution of messages from a source to its clients. In each case, one must address the issue of the *optimal configuration* of the hierarchy: Assuming a given number of items, how to choose optimally the number of levels in the hierarchy (and thus the number of items at each level). Without loss of generality, we formalize this problem in the domain of assembly or manufacturing. We consider the process of manufacturing a product from a given number of elementary components. By assembling intermediate products, the target product can be manufactured in a variety of processes, each modeled by a tree. We are interested in *manufacturing turnaround*: the time between receiving an order at the root and its completion. We express the turnaround time of each manufacturing process (tree) with a formula that incorporates three parameters: the time required to *create* elementary components, the time required to *assemble* a product from its components and the time required to *deliver* the product to its procurer (another manufacturer). We show that this turnaround formula is optimized in a manufacturing process that corresponds to a *perfect* (or *nearly perfect*) tree. Somewhat surprisingly, the degree of the optimal tree (i.e., the ideal number of components in each sub-assembly) is shown to be independent of the number of elementary components, suggesting that in each manufacturing environment there is an ideal assembly size, which is optimal for the manufacturing of products of any scale.

## 1 Introduction

Hierarchy theory suggests that hierarchical organization is a preferred approach to many complex systems [3]. In particular, it is common to model complex

manufacturing and assembly processes with trees. In these trees, each node is a manufacturing or assembly step and an edge between two nodes indicates that the product represented by the lower level node is used in the manufacturing of the product represented by the higher level node. A leaf node corresponds to the manufacturing of an *elementary product*, a product that incorporates no other products (a product made from scratch).<sup>1</sup> An internal node corresponds to the manufacturing of a *composite* product, a product that incorporates the products modeled by its children nodes; products modeled by internal nodes are procured by and delivered to other manufacturers. The root node corresponds to the manufacturing of the *final* product, which is composite as well.

Obviously, each manufacturing tree has a prescribed number of leaves (the elementary products required to create the final product), and a single root node where the final assembly occurs. Other than that, the structure is flexible. A shallow manufacturing tree implies few intermediate manufacturing steps, each involving a large number of components. Conversely, a deep manufacturing tree implies many intermediate manufacturing steps, each involving a small number of components.

We are interested in *manufacturing turnaround*: the time between receiving an order at the root and its completion. We identify three major time components in any manufacturing process: (1) The time required to create an elementary product from scratch; it is assumed to be the same for all elementary products; (2) the time required to *assemble* a product from a given set of component products; it is a monotonic function of the number of components used; and (3) the time required to *deliver* a product from one manufacturer to another; it is assumed to be fixed for all products. The last two times are at odds with each other. High assembly times give advantage to assemblies that involve few components (and hence to deep manufacturing trees); high delivery times give advantage to manufacturing processes that involve few intermediate components (and hence to shallow manufacturing trees). Based on these time components we define a turnaround function for manufacturing trees.

The subject of this paper is finding the *optimal manufacturing tree* (from the perspective of turnaround) for a given product in a given manufacturing environment. More specifically, given the times of creation, assembly and delivery, what are the optimal assembly *breadth* (number of components per intermediate products) and *depth* (number of levels in the manufacturing hierarchy) to manufacture a product that incorporates a given number of elementary components.

Although stated here in terms of manufacturing or assembly processes, the problem is more general. It applies in any architecture that incorporates a set of items in a tree-like hierarchy, with an intrinsic conflict between the costs associated with the depth and the breadth of the hierarchy. We mention here three different examples.

**File and menu organization, taxonomies, hierarchical clustering.** The ideal organization of a set of files in a folder hierarchy is easily formulated in

---

<sup>1</sup> Additionally, in a manufacturing environment that imports some of its components, an imported component is considered elementary as well.

term of the manufacturing trees discussed in this paper. Locating a file in a folder hierarchy involves two times: the time to search a folder for the correct item (equivalent of “assembly time”), and the time to open a subfolder and display its content (equivalent of “delivery time”). High search times reflect the difficulty in searching densely-populated folders; they encourage small folders and deep hierarchies. High traversal times reflect the overhead of changing folders and refreshing the display; they encourage large folders and shallow hierarchies. The overall cost associated with a file hierarchy is measured by the costliest path from the root folder to a file. A well-known issue which is equivalent to file organization is the organization of menus and sub-menus in user interfaces [2]. Similar problems also exist in taxonomies and hierarchical clustering.

**Message distribution.** Many systems for distribution messages or packets assume a single message source and a hierarchy of “switches” to forward the message to a set of clients. A switch that forwards a message to a large number of recipients incurs higher costs (for example, the message may spend more time in the switch, or costlier hardware may be required); on the other hand, each level in the distribution hierarchy incurs additional delay and possibly decreased reliability. The overall cost to be associated with a distribution system measures the costliest path from the source to a client, and the optimization finds the ideal number of “ports” on every switch. Similar situations often exist in systems that distribute actual goods.

**Organizational hierarchy.** Large enterprises are often concerned with the efficiency of their organization. The “leaves” of an enterprise are its indivisible operational units, and the issue is the proper level of branching in intermediate organizational units (often referred to as *span of control* [1]) and the proper length of the chain of command from the command center to the basic operational units (often referred to as *levels of management* [5]). Each level in the command incurs delays, bureaucratic overhead, increased inefficiency, and potential loss of control. On the other hand, intermediate units with large numbers of subordinates incur other types of inefficiencies, backlogs, and various managerial problems. The optimal organizational hierarchy states, for a given number of basic operational units, the ideal size of intermediate units and hence the overall depth of the organization. Quantifying the costs of “nodes” and “edges” undoubtedly presents a challenge.

In Section 3 we show that for every product there is an optimal manufacturing tree which is *perfect* or *nearly perfect*. Perfect trees are trees that are both balanced (all leaves are equal distance from the root) and full (all non-leaf nodes have the same degree  $d$ ). Nearly perfect trees are balanced and full, but their non-leaf nodes are of two consecutive degrees ( $d$  and  $d + 1$ ), with nodes at the same level having the same degree.

In a given manufacturing environment, a product with a given number of elementary components may be assembled by different perfect or nearly perfect trees, each with its own degree. The search for an optimal manufacturing process has therefore become a search for the optimal degree (assembly size). In Section 4 we show how to select the optimal tree among these alternatives. We also show

that the degree of the optimal tree depends only on the ratio of the assembly and delivery times; surprisingly, it does not depend on the number of leaves (elementary components).

We begin with a description of the manufacturing model, and a brief review of tree terminology.

## 2 Modeling Manufacturing Processes with Trees

We assume that manufacturing processes are modeled with trees, as described in the introduction. In practice, a manufacturing step may require only a single component; i.e., it may simply refine and add value to one component. However, we assume here that each manufacturing step requires at least two components. Without this assumption, the depth of the manufacturing tree for a product of  $n$  elementary components is not bounded, as it could involve an unlimited number of such single component refinements.

Denoting with  $n$  the number of elementary components, at one extreme is a tree of depth 1, that models a manufacturing process in which the final product is manufactured from its  $n$  elementary components in a single manufacturing step. At the other extreme is a tree of depth  $\lceil \log_2 n \rceil$ , that models a process in which each manufacturing step combines exactly two components (the required minimal number) in a new product.

### 2.1 The Cost Model

In comparing these alternative processes for manufacturing a specific product from its  $n$  elementary components, our focus is on the *time* it takes to manufacture (deliver) the final product. This turnaround time is composed of three different times:

1. **Creation time.** This is the time spent at leaf nodes. It models the time required to create an elementary product from scratch.
2. **Assembly time.** This is the time spent at each non-leaf node. It models the time required to assemble a product from its components.
3. **Delivery time.** This is the a time spent at each edge. It models the time required to deliver an intermediate product from one manufacturer to another.

We assume that assembly time increases with the number of components (the number of children nodes). This monotonicity assumption models the indisputable fact that assembling a large number of components requires more time than assembling a small number of components. This is true whether the components are Lego bricks or vegetables to be peeled and chopped for a stew.

Specifically, we assume that assembly time is a *convex* function of the number of components. That is, denote the number of components  $x$  and the assembly time  $f(x)$ , then for any two points  $x_1, x_2$  and  $0 \leq \alpha \leq 1$ ,  $f(\alpha \cdot x_1 + (1 - \alpha) \cdot x_2) \leq \alpha \cdot f(x_1) + (1 - \alpha) \cdot f(x_2)$ . Intuitively, a convex function is a continuous function



whose value at the midpoint of every interval does not exceed the average of its values at the ends of the interval. Linear or quadratic functions are convex. We assume that the same function  $f$  is associated with every non-leaf node in the manufacturing tree.

An example of linear assembly time is when each of the  $x$  components in the assembly requires a single action, and the time for each action is  $a$ . Then  $f(x) = a \cdot x$ . An example of quadratic assembly time is when each *pair* of the  $x$  components requires an action and the time for each action is  $a$ . Then  $f(x) = a \cdot x^2$ .

Our manufacturers do not keep any stock of components necessary for manufacturing. If a manufacturer receives a request for a product, it must order the components needed to manufacture that product.<sup>2</sup> Therefore, the use of intermediate products is more time-consuming as it initiates a chain of orders that terminates in the elementary products. We assume that all delivery times are identical; that is, a single time  $b$  is associated with every edge in the manufacturing tree.

Whereas assembly times encourage deep manufacturing trees with few components per product, delivery times encourage the opposite: shallow manufacturing trees with many components per product. Finally, we assume that all creation times are identical; that is, the same time  $c$  is required to create each of the  $n$  elementary components.

Consider now a path in the manufacturing tree, from the root to a leaf node. Denote this path  $v_0, v_1, \dots, v_h$ , where  $v_0$  is the root and  $v_h$  is a leaf. Let  $d_i$  denote the number of children of node  $v_i$  ( $0 \leq i \leq h-1$ ). We associate an overall time with this manufacturing path: the creation time at the leaf plus the assembly times at each intermediate node, plus the delivery times along the edges of the path, plus the final assembly time at the root.

**Definition 1.** The *lag* of a manufacturing path  $v_0, v_1, \dots, v_h$  from the root  $v_0$  to a leaf  $v_h$  is

$$\sum_{i=0}^{h-1} f(d_i) + h \cdot b + c$$

Since manufacturing begins at the leaves and proceeds along multiple paths simultaneously towards the root, completion time is governed by the slowest manufacturing path.

**Definition 2.** The *turnaround* of a manufacturing tree is the highest lag of any of its manufacturing paths.

In calculating turnaround, we only measure the time for assembling and delivering products in the upward direction, ignoring the time required to propagate orders in the downward direction. Clearly, this should not impact the search for optimal trees; as ordering time along an edge may be assumed to be embedded in the delivery time  $b$ .

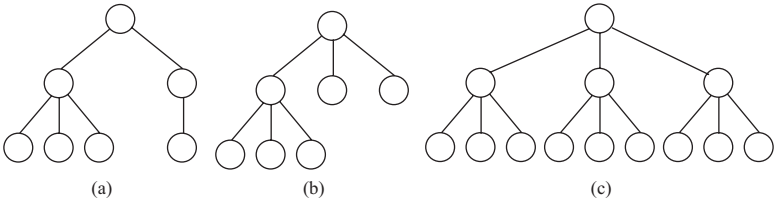
---

<sup>2</sup> Such recursive processes are sometimes referred to as *lazy*.

### 2.2 Tree Terminology

The terminology we use is standard (for example, [4]). The *length* of a path in a tree is the number of edges in the path. The *height* of a tree is the length of its longest path from the root (thus the height of a tree which is only a single node is 0, and the height of tree which is a root node connected to a set of leaf nodes is 1). The *level* of a node is the length of the path from that node to the root (thus the level of the root is 0).

A tree is *balanced* if all the paths from the root to a leaf have the same length. A tree is *full* (of degree  $d$ ) if all its non-leaf nodes have  $d$  children. Balanced and full binary trees are sometimes referred to as *perfect* trees; hence, we refer to balanced and full trees (of degree  $d$ ) as *perfect* trees (of degree  $d$ ). Figure 1 shows a tree which is balanced but not full (a), a tree which is full (of degree 3) but not balanced (b), and a tree which is perfect (of degree 3) (c).

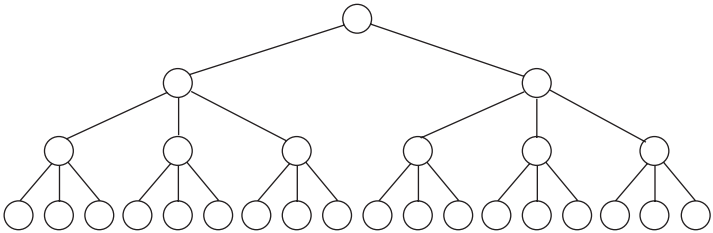


**Fig. 1.** Balanced, full and perfect trees

When a tree is perfect, its height  $h$ , degree  $d$ , and number of leaves  $n$  maintain:  $n = d^h$ . Or, alternatively,  $h = \log_d n$  or  $d = n^{1/h}$ .

In general, when  $n$  and  $h$  are given, there may not be an integer  $d$  that satisfies this relationship. In such cases we approximate a perfect tree with a tree of the following properties:

1. The degree of non-leaf nodes is either  $d$  or  $d + 1$ .
2. Nodes at the same level have the same degree.



**Fig. 2.** Nearly perfect tree of degree 2-3

We refer to such trees as *nearly perfect* trees of degree  $d - (d + 1)$ . Figure 2 shows a nearly perfect tree of degree 2-3. Note that the second property implies that the tree is balanced. Also note that there are no conditions on the number of levels of each type, or their order.

### 3 There Is Always a (Nearly) Perfect Optimal Tree

Let  $T$  be a manufacturing tree in the manufacturing environment defined earlier. Let  $h$  be its height, let  $n$  be its number of leaves, and let  $d_i$  be the degree of node  $v_i$ . Recall that in this environment, the time spent at a non-leaf node  $v_i$  is a convex function  $f(d_i)$ , the time spent at every edge is  $b$ , and the time spent at every leaf node is  $c$ . The lag of a path  $v_0, v_1, \dots, v_h$  from the root  $v_0$  to a leaf  $v_h$  is  $\sum_{i=0}^{h-1} f(d_i) + h \cdot b + c$ , and the turnaround of  $T$  is the highest lag of any of its root-to-leaf paths.

In a manufacturing environment with costs  $f$ ,  $b$  and  $c$ , consider a product that requires  $n$  elementary components. Let  $T$  be a manufacturing tree that optimizes the turnaround time, and let  $\epsilon$  denote the optimal turnaround time. That is, the slowest manufacturing path has lag  $\epsilon$ . Note that there could be several root-to-leaf paths of this lag.

Consider the top-level subtrees of  $T$  (the subtrees rooted at level 1 nodes). Let  $T_s$  be the top-level subtree with the most leaves (if several subtrees share this distinction, then one of these is chosen at random, and if all subtrees have the same number of leaves, then a subtree is chosen at random).

We observe that  $T_s$  includes a root-to-leaf path whose lag is  $\epsilon$ . Otherwise, we would create a new tree by replacing every top-level subtree of  $T$  with  $T_s$ . The number of leaves of the new tree would be at least  $n$ , but its turnaround would be strictly lower than  $\epsilon$ , in contradiction with the optimality of  $T$ .

We now create a new tree  $T'$  by replacing every top-level subtree of  $T$  with  $T_s$ . Since  $T_s$  includes a root-to-leaf path of lag  $\epsilon$ , the new tree retains the same turnaround  $\epsilon$ . Since  $T_s$  has the most leaves, the number of leaves of the new tree is at least  $n$  (the number of leaves is  $n$  only if all top-level subtrees of  $T$  have the same number of leaves).

Consider now the identical subtrees of  $T'$ . Repeating the same construction, we replace each of these subtrees with a tree whose top-level subtrees are identical. Again, the new tree has at least as many leaves as the old tree, but it retains the same turnaround. We work our way down in this way until level  $h - 1$  subtrees (in these final subtrees the roots are directly connected to the leaves).

Eventually,  $T$  is transformed to a new tree  $T^*$ , in which all the subtrees that are rooted at the same level are identical. Consequently, the degrees of nodes at the same level are identical, and hence all root-to-leaf paths have the very same length and lag. Note that the turnaround of  $T^*$  remains  $\epsilon$ , and its number of leaves is at least  $n$ .

Recall that in  $T^*$ , nodes at the same level have the same degree. Let  $d_i$  denote the degree of nodes at level  $i$  ( $0 \leq i \leq h - 1$ ; we ignore the leaf level, whose

degree is  $d_h = 0$ ). Assume that among these, there are degrees that are different by more than 1; i.e., assume the tree has levels  $i$  and  $j$ , such as  $d_i - d_j > 1$ .

We transform  $T^*$  to another tree. The new tree is identical to  $T^*$ , except that we replace the degrees at level  $i$  and  $j$  with new degrees  $d'_i = d_i - 1$  and  $d'_j = j + 1$  (if  $d_i - d_j = 2$ , then  $d'_i = d'_j$ ). The following inequalities are easy to verify:

1.  $f(d'_i) + f(d'_j) \leq f(d_i) + f(d_j)$
2.  $d'_i \cdot d'_j > d_i \cdot d_j$

The first inequality is due to the convexity of  $f$  and it implies that the turnaround of the new tree is at least as good as that of the old tree. The second inequality implies that the new tree has strictly more leaves. If the new tree still has two levels whose degrees are different by more than 1 (for example,  $d'_i$  and  $d'_j$  themselves may still be different by more than 1), then the same construction is repeated.

Eventually,  $T^*$  is transformed to a new tree  $T^{**}$ , in which the degrees of nodes at the same level are identical, and the degrees of non-leaf nodes are different by at most 1. The latter fact implies that non-leaf nodes are either of a single degree  $d$  or of two degrees  $d$  and  $d + 1$ . Recall that  $T$  achieved optimal turnaround  $\epsilon$  for  $n$  elementary components (leaves). Throughout its transformation to  $T^{**}$ , the number of leaves has been at least  $n$ , and the turnaround has been at most  $\epsilon$ . Because of the optimality of  $T$ , it is not possible that  $\epsilon$  has decreased, but it is possible that  $n$  has increased, as small increases in the number of leaves are possible that do not affect the turnaround.

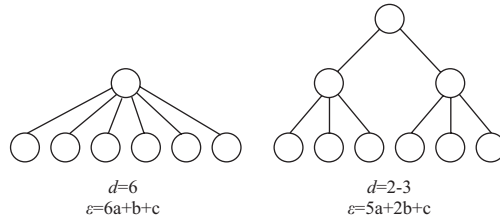
This construction proves the following theorem.

**Theorem.** For every product, there exists an optimal tree which is either perfect or nearly perfect.

Note that the optimal tree may include extra leaves at no additional cost; excess leaves can be removed arbitrarily from the perfect or nearly perfect tree, without affecting the turnaround, thus obtaining an optimal tree for the given number of elementary components (the resulting tree is no longer perfect or nearly perfect, because its bottom level is no longer full).

## 4 Finding Optimal Trees

Consider the same manufacturing environment and a product that requires  $n$  elementary components. From the theorem we know that there is a perfect or a nearly perfect manufacturing tree that optimizes the manufacturing turnaround. However, multiple such trees are possible, each with a different turnaround. For example, two alternative trees for  $n = 6$ , one perfect and one nearly perfect, are shown in Figure 3, with their specific turnaround formulas (for simplicity, the example assumes linear assembly time:  $f(d) = d \cdot a$ ). For  $a = 10$  and  $b = 1$  the right tree would give shorter turnaround, whereas for  $a = 1$  and  $b = 10$  the left tree would be preferred. Intuitively, when assembly time is low and delivery time is high, a shallow manufacturing tree with a large degree would be preferred;



**Fig. 3.** A perfect and a nearly perfect tree for  $n = 6$

conversely, when the assembly time is high and the delivery time is low, a deep tree with a small degree would be preferred. A brute force solution for finding the optimal tree is to enumerate all the possible perfect or nearly perfect trees for a given product and evaluate their turnarounds.

In this section we consider the issue of selecting the optimal tree from the possible alternatives. Our treatment, however, is *approximative*.

We treat the manufacturing tree as a continuous structure, not the discrete structure that it is. In our model, the manufacturing of a given product is modeled by a tree. In the previous treatment, the parameters of such trees (e.g., their height, node degrees and total number of leaves) were integers. In this section, we relax this restriction: We allow node degrees that are not necessarily integers (while maintaining the number of leaves and the height as integers).

In other words, we define a slightly different optimization problem. Assume a convex function  $f$ , numbers  $b$  and  $c$ , and integer  $n$ . Assume a sequence of numbers  $d_0, \dots, d_{h-1}$ , such that  $n = \prod_{i=0}^{h-1} d_i$ , and consider the cost function  $t(d_0, \dots, d_{h-1}) = \sum_{i=0}^{h-1} f(d_i) + h \cdot b + c$ . This problem corresponds to the tree  $T^*$  from the previous section: a balanced tree in which nodes at the same level have the same degree.

Note that both the length of the sequence and the individual values in the sequence are variables; however, the product of the entire sequence is constant. By convexity of  $f$ , we get that for each value  $h$ , the cost function  $t$  is minimal when the  $d_i$  are identical:  $d_i = n^{1/h}$ . This suggests that the optimal structure is a perfect “tree” of degree  $n^{1/h}$ , but note that this degree is not necessarily an integer.<sup>3</sup>

Note that by relaxing the requirement that the degree be an integer, a perfect “tree” can be obtained for any number of leaves  $n$  (previously, the perfect or nearly perfect tree often had more leaves than the given  $n$ ).

In the new perfect “tree”, the time spent at each node is always  $f(d)$  and the length of every root-to-leaf path is  $h$ . Hence, the turnaround is simply  $f(d) \cdot h + b \cdot h + c$ . In this expression,  $h$  and  $d$  are variables, but since  $h = \log_d n$ , turnaround can be expressed as a function of  $d$  only (with constants  $n$ ,  $b$ , and  $c$  and the convex function  $f$ ):

<sup>3</sup> This suggests why a nearly perfect tree has two consecutive degrees: these are the integers “around”  $d$ .

$$\begin{aligned} t(d) &= f(d) \cdot \log_d n + b \cdot \log_d n + c \\ &= \log_d n (f(d) + b) + c \end{aligned}$$

Note that  $t(d)$  is defined only for  $d \leq n$ .

The search for an optimal manufacturing tree has now become a search for the optimal degree  $d$ . Of course, a non-integer value of  $d$  cannot be used in a manufacturing process, but it should serve as a good approximation of the “true” degree of the optimal process.

For specific analyses, we consider two individual convex assembly functions.

#### 4.1 Linear Assembly Time

Assume first that assembly time is a linear function; i.e.,  $f(x) = a \cdot x$ , where  $x$  is the number of components and  $a$  is a constant per-component assembly time. Such functions model assemblies in which every component requires a single action and the time for each action is  $a$ . The turnaround formula becomes

$$t(d) = \log_d n \cdot (a \cdot d + b) + c$$

Figure 4 charts two examples of this turnaround function. In both cases the number of elementary components (leaves) is  $n = 500$ , and creation time is  $c = 5$ . In one case assembly time is higher than delivery time ( $a = 2$  and  $b = 1$ ), in the other case delivery time is higher than assembly time ( $a = 1$  and  $b = 10$ ).

The optimum value of this turnaround function may be found by solving the equation  $t'(d) = 0$ . We get this expression for the optimal  $d$ :

$$d \cdot \ln d - d = b/a$$

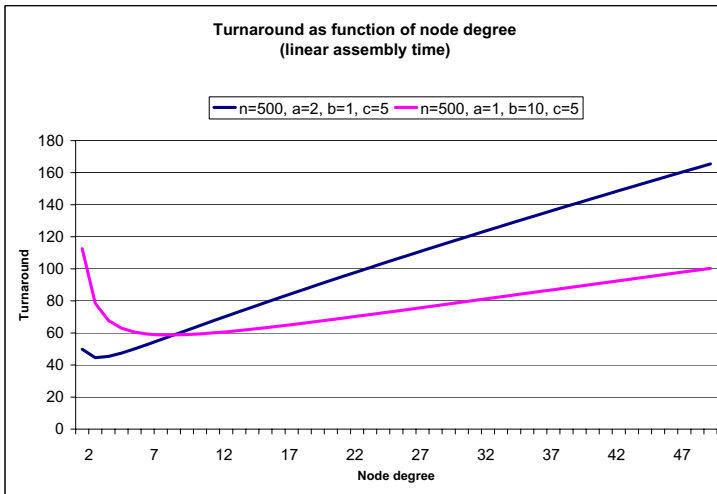
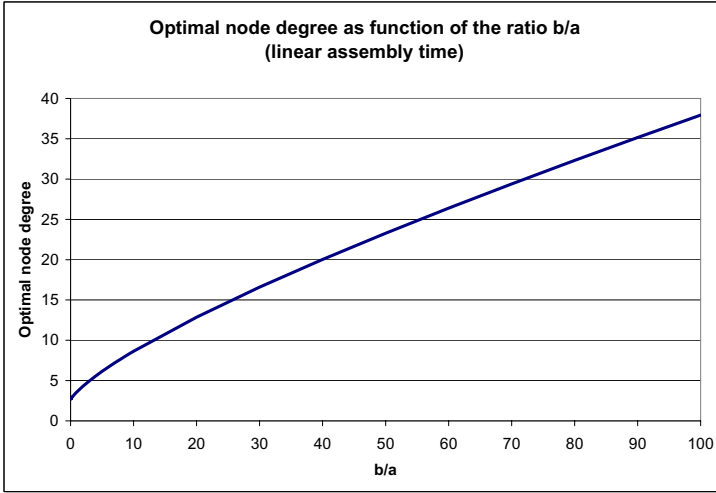


Fig. 4. Turnaround as function of node degree (linear assembly time)



**Fig. 5.** Optimal node degree as function of the ratio  $b/a$  (linear assembly time)

While this equation does not have an open solution for  $d$ , it can be approximated with common techniques.

The equation reveals an interesting fact: *the optimal solution is independent of  $n$* . Of the four parameters of the problem,  $n$ ,  $a$ ,  $b$ , and  $c$ , the optimal manufacturing “tree” is of a degree that depends only on the ratio  $b/a$ . This somewhat surprising conclusion implies that for each manufacturing environment, there is an *ideal assembly size*, which is optimal for the manufacturing of products of *any* scale (i.e., any number of elementary components).

Figure 5 charts the optimal degree as a function of the ratio  $b/a$ . This graph can be used to estimate the optimal assembly size in different manufacturing environments.

## 4.2 Quadratic Assembly Time

Assume now that assembly time is a quadratic function; i.e.,  $f(x) = a \cdot x^2$ , where  $x$  is the number of components and  $a$  is a constant per-component assembly time. Such functions model assemblies in which every pair of components requires an action and the time for each action is  $a$ . The turnaround function becomes

$$t(d) = \log_d n \cdot (a \cdot d^2 + b) + c$$

Two examples of this turnaround function are charted in Figure 6. In both cases the number of elementary components (leaves) is  $n = 500$  and the creation time is  $c = 5$ . In one case assembly time is higher than delivery time ( $a = 2$  and  $b = 1$ ), in the other case delivery time is higher than assembly time ( $a = 1$  and  $b = 1000$ ).

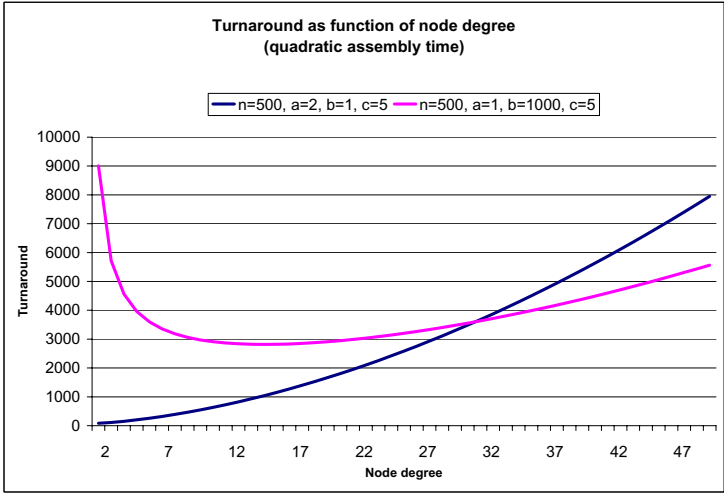


Fig. 6. Turnaround as function of node degree (quadratic assembly time)

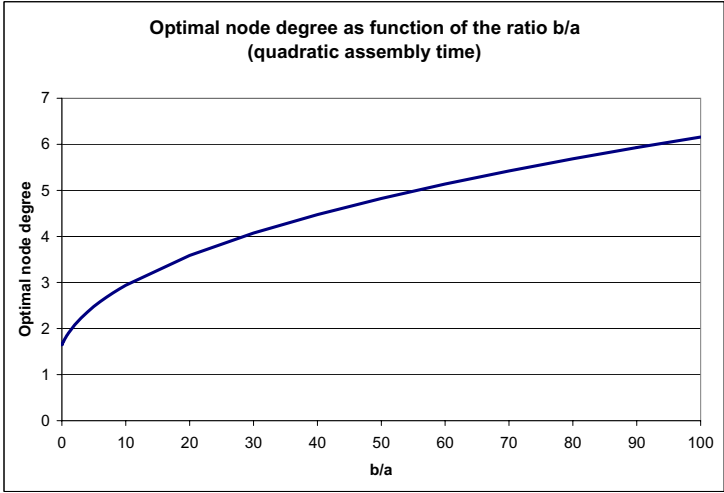


Fig. 7. Optimal node degree as function of the ratio  $b/a$  (quadratic assembly time)

The equation  $t'(d) = 0$  provides an expression for the optimal  $d$ :

$$2d^2 \cdot \ln d - d^2 = b/a$$

Once more, we observe that the degree of the optimal manufacturing “tree” does not depend on the number of leaves (elementary components), but only on the ratio  $b/a$  of manufacturing environment parameters. Figure 7 charts the optimum degree as a function of this ratio.



## 5 Conclusion

Hierarchical organization is a standard approach in a wide variety of complex systems. While the issue of breadth *vs.* depth of hierarchies has been raised in some of these contexts, the authors are not aware of any prior theoretical treatment of this trade-off.

The process of manufacturing (assembly) of complex products from elementary products provides a good context for a theoretical analysis of this trade-off, and in this paper we adopted its terminology and objectives (but recall that in Section 1 we showed other contexts to which the analysis is directly applicable).

We assumed that manufacturing environments are specified with three parameters: the time for creating elementary components, the time for assembling partial products, and the time for delivering partial products to their procurers. Each manufacturing process was then assigned a cost that corresponds to *turnaround*: the time between the receiving of a manufacturing order and its completion. In this model, we addressed the problem of *optimal manufacturing processes*: Given a product with a specified number of elementary components, find a manufacturing process that optimizes the turnaround. This problem translates immediately to finding the optimal *assembly size* for a given product.

We proved that for a given number of elementary components there exists an optimal tree which is perfect or nearly perfect. Since there are several such trees for a given number of elementary components, we showed how to find the optimal tree among these by means of simple differentiation. We observed that the optimal degree (optimal assembly size), for assembly times that are linear or quadratic in the number of components, is a function of the ratio of the assembly and delivery times only (and is independent of the number of components). This somewhat surprising conclusion implies that in each manufacturing environment there is an *ideal assembly size*, which is optimal for the manufacturing of products of *any* scale (i.e., any number of elementary components).

Our model is simple, in that its three cost parameters are uniform across the manufacturing environment. That is, all products have identical creation and delivery times, there is a single function that calculates the cost of assembly, and it depends only on the number of components in the assembly (not on the specific components). We are interested in extending the model to consider manufacturing environments in which different products (or categories of products) incur different costs.

Another underlying assumption is that each manufacturing node is capable of assembling any combination of products. It would be interesting to consider a more general model in which various types of constraints must be enforced; for example, certain products cannot be combined with certain other products in a single sub-assembly, certain products must be combined with certain other products in any sub-assembly, some nodes have limits on the size of their assemblies, and so on.

## References

1. L. Gulick. Notes on the theory of organization. In L. Gulick and L. Urwich, editors, *Papers on the Science of Administration*, pages 191–195. Institute of Public Administration, Columbia University, New York, 1937.
2. B. Schneiderman. *Designing the User Interface: Strategies for Effective Computer Interaction*. Addison-Wesley, 1992.
3. H.A. Simon. The organization of complex systems. In H.H. Pattee, editor, *Hierarchy Theory*. George Braziller, New York, 1973.
4. D.B. West. *Introduction to Graph Theory*. Prentice Hall, 2001.
5. J. Woodward. *Industrial Organization: Theory and Practice*. Oxford University Press, 1965.

# A Spreadsheet Client for Web Applications

Dirk Draheim<sup>1</sup>, Peter Thiemann<sup>2</sup>, and Gerald Weber<sup>3</sup>

<sup>1</sup> Institute of Computer Science, Universität Mannheim  
`draheim@acm.org`

<sup>2</sup> Institute of Computer Science, Universität Freiburg  
`thiemann@acm.org`

<sup>3</sup> Dept. of Computer Science, University of Auckland  
`g.weber@cs.auckland.ac.nz`

**Abstract.** There is an increasing gap between web services and web applications. While web services communicate via typed interfaces (e.g., using WSDL and SOAP), many web applications still rely on untyped, manually programmed forms in the restricted HTML widget set. These limitations cause developers to resort to HTML with client-side scripting, resulting in applications that can be hard to maintain. The goal of our work is to close the gap and ease maintenance by providing a browser technology that relies on declarative specifications and supports a fully typed interaction model. To this end, we have implemented SPREANS, a type-aware spreadsheet that edits hierarchical data, supports a rich widget set, and includes browser functionality by implementing an interface to web services. The use of SPREANS as a browser replaces the main uses of client-side scripting (calculations, constraint checking, and UI embellishment) by declarative specification, thus simplifying maintenance.

## 1 Introduction

With the advent of web services, a new generation of web-enabled information systems is foreseeable. Compared to current interactive web applications, web services focus on content over presentation and there are standards that govern message formats (SOAP/XMLP), the type structure of messages (WSDL), the coercion of services (WSCDL), and so on. XML technologies like XPath, XSLT, and XQuery are available for further processing, transformation, and querying of the results. That is, web services inhabit a typed world with declarative specifications and transformations that enable straightforward interfacing with the application logic. Some of this interfacing can be (and has been) automated by generating code from the specifications.

How can the development of end-user web applications, like enterprise applications, take advantage of the new developments? In this area, the so-called web architecture is well established and successful. Its user-interface layer consists of just the web browser, so it is an example of an ultra-thin-client architecture. However, enterprise applications often have a demand for advanced functionality that goes beyond the offerings of plain HTML forms as implemented in the current generation of web browsers. These browsers consider a form as a means for

entering a list of strings associated with a static set of tags, which is predefined in the form. This consideration is limiting with respect to (i) type validation for forms and (ii) application-specific widget sets.

Regarding type validation, the server's validation of form submissions and their conversion to proper types must be implemented manually. Sometimes, another layer of client-side validation is provided using a scripting language like JavaScript. Client-side validation avoids additional communication overhead in case of type errors, but raises the additional problem of keeping the two validation procedures in sync.

Regarding widgets, web browsers support the input widgets of HTML forms, which are generally perceived as limiting. As an example, consider entering tabular data in an enterprise system. A user would like to add/delete rows (rarely columns) as well as edit existing rows. HTML forms neither provide a suitable widget for this frequent task, nor do they do provide the required dynamic tag set. Hence, the developer has to resort either to server-side frameworks, like the user interface components of Jakarta Struts or JavaServer Faces, or to scripting technology on the client-side, with the same tradeoffs as for validation.

To avoid the maintenance problems caused by heavy use of scripting and to also avoid the sometimes tricky development of scripts that have to run on every browser, our approach is to follow the lead of web services by creating a browser that works from declarative specifications of form types and messages, and that builds directly on a rich widget set. Furthermore, we base our browser on the spreadsheet paradigm so that client-side computations – like computing totals or taxes interactively during user input – can be specified declaratively by formulae and kept automatically up-to-date. Thus our browser eliminates the most frequent reasons for using client-side scripting from the start.

Our browser furthermore enforces the separation of content and layout by moving from HTML to XML and web services. The reliance on XML has the usual advantages that simplify the development: XML Schema is the basis for the type structure and the XML tool chain (parsing, Schema validation, navigation, transformation) is readily available. Note that we are using XML in a data-centric way, the semistructured nature of XML is not of interest here.

The contributions of this work are as follows.

- A novel browser-enabled spreadsheet for form-based applications. The novelty is in the combination of the following aspects.
  - Integration into a fully typed client-server interaction model.
  - Spreadsheet editing of an underlying typed XML data structure.
  - User interface with configurable widgets instead of the traditional grid.
  - Functionality to submit forms through web services.
  - Facility for web service browsing.
- A typed, declarative client-server protocol for enterprise applications. The protocol implements the ideas of form-oriented analysis [6] using XML.

- Separation of content and layout of a form. The central specifier of a form is its type. The data contents of the form and its layout are separate components that only depend on the type.
- Specification of restrictions and client-side computation with constraints.

SPREANS is the prototype implementation of such a spreadsheet browser<sup>1</sup>. SPREANS is available for download at <http://spreans.formcharts.org/>. The download package includes a worked out example application (a bookshop). Section 4 contains some excerpts from that example.

Section 2 reviews the foundations of form-based interactive systems. Next, Section 3 explains the architecture of the spreadsheet browser in general terms before Section 4 delves into the specifics of the modeling of a response-page specification using various XML technologies with particular emphasis on the modeling of types and constraints. Section 5 gives an overview of the technical aspects of the implementation. Finally, Section 6 discusses related work and Section 7 concludes.

## 2 From Forms to Spreadsheets

Web applications are recent examples of *form-based applications* that interact with users through a sequence of HTML forms. Enterprise applications like ERP products, web portals, and traditional mainframe-terminal applications typically have form-based user interfaces, too. The *submit-response interaction style* [6] provides a technology independent characterization of this kind of interaction.

A submit-response interaction has two stages. In the first, *client-page* stage, the user enters data through a form. This interaction is usually transient and non-modal, *i.e.*, a user may edit form elements in arbitrary order. The entry becomes effective through the second *server-action* stage of interaction, experienced as a *page change*. A page change is an atomic action that commits the input data by submitting a message to the application. This *user message* triggers a *server action* that produces another message, a *response page*.

A *form chart* is a bipartite state machine for modeling submit-response interaction. The machine alternates between the two kinds of states corresponding to the client-page and server-action stages. The form chart also specifies types for all messages between client and server. This system view yields a precise model of form-based interaction that elides layout considerations and abstracts from platform specifics.

Turning to implementing form-based system as web applications, the customary HTML forms have a number of shortcomings. They lack support for types, dynamically changing widgets, and for performing client-side computation and validation. Only low level scripting can be used to rectify these shortcomings. Furthermore, web applications that deliver HTML restrict themselves to end-users because they combine content and layout already on the server. With the present focus shift to a service-oriented architecture, the separation of content

---

<sup>1</sup> Thanks to Matthias Bild for implementing the system as part of his Master's thesis.

and layout becomes a *conditio sine qua non* for ensuring system integration and reusability of services. Hence, the separation should be maintained as far down the pipeline as possible.

The SPREANS browser embeds form processing in a spreadsheet and provides liberal means for controlling its layout. It requires separate specifications for content and layout (*i.e.*, model and view). This separation supports fully typed data and message models. Finally, the spreadsheet-style widgets are naturally dynamic and provide for computation and validation.

We have chosen a spreadsheet because it is a proven end-user-oriented technology [10, 8]. However, we had to change the editing model from the traditional infinite grid of cells to a hierarchical (XML) data structure with grid coordinates replaced by XPath expressions, similarly to the XForms proposal [7].

### 3 The Spreadsheet Browser Software Architecture

This section explains the architecture of SPREANS, our spreadsheet-based browser. One important point of the design is that it consequently avoids points which may require scripting in the standard-browser approach. These points are the extension of the widget set and the client-side validation of data.

The main architectural components of a simple standard HTML web browser are a GUI, a rendering engine, a document manager, and a URL resolver. The purpose of the GUI and the rendering engine for HTML should be obvious. The URL resolver stands for the mapping from URL to document. It contains subcomponents for accessing URLs with various protocols and for caching of documents. The document manager has subcomponents for keeping track of history, bookmarks, and so on. It also keeps track of data entered into forms and is responsible for sending submitted data with a URL to the URL resolver which transforms it into a HTTP message and sends it to the web server. The core responsibility of the document manager is as follows. On the first navigation to a web page, the document manager receives a URL from the GUI, retrieves the corresponding document using the URL resolver, and hands it off to the rendering engine for display. A subsequent navigation by clicking a link is quite similar, whereas a navigation by clicking a submit button requires the document manager to first retrieve the data entered into the document's form fields before passing the URL (with the data) to the URL resolver.

The main novelty of a spreadsheet-based browser lies in its document manager. It comprises additional subcomponents for type validation, constraint validation, constraint evaluation, and query extraction. To understand the tasks of these subcomponents, we consider the processing of a query.

When pointing SPREANS to the URL of a service, it receives a response with five logical parts: the type of the content data, constraints on the content data, the initial content data of the page, a list of server-action bindings, and the layout of the page. The term “content data” comprises all the variable data computed by the server (the initial content data) and the data that a user may

later enter into the page. A server-action binding consists of a target URL and an XSLT stylesheet that extracts the query data from the content data.

Compared to the standard browser, there are three major differences. First, all data is typed and the browser is aware of the types. The browser validates each message so that the server only receives well-typed messages. Second, all further parts—including the layout—depend only on the type of the data. Third, the layout is strictly separated from the content data. It contains pointers into the content data to enable the browser to transfer content data to and from the rendered page. These pointers are valid for all data values of the specified type.

The choice of separating data and metadata in this way has a number of interesting implications. First, types, constraints, server-action bindings, and layouts may be represented by URLs (*i.e.*, pointers) so that they can be loaded, cached, and shared separately from the actual content data. Second, if a page layout is not available, our implementation includes a generation tool that transforms the provided type specification to a standard layout.

A distinguishing feature of a spreadsheet is its ability to compute dependent parts of the spreadsheet from independent ones. In SPREANS, the type specification includes information as to which data item is to be computed from others. The actual value of a computed item is determined by a constraint formula, which can be evaluated once all its input prerequisites are available. Hence, one user modification to the content data may lead to many transitively dependent changes in the content data. These changes are also reflected in the display. Contrary to a standard spreadsheet, end users cannot enter their own formulae or modify existing ones because the evaluation constraints are part of the specification of the application.

We call the responsible subcomponent “constraint evaluation” as opposed to “constraint validation” because it causes different effects (changing a dependent value as opposed to rejection of a modification) and also because it is implemented by different means.

Finally, clicking a particular submit button on the page selects a server-action binding, that is, a target URL and a projection. The document manager extracts the query data from the current content data, sends the result to the target URL, and awaits a response page.

The rendering engine is another issue. A spreadsheet-based browser requires a rendering engine that supports a more sophisticated set of widgets than an HTML rendering engine. For example, the basic widget of a spreadsheet is an editable table that supports operations for adding rows and columns. While such a widget could be created on top of HTML with some effort and a lot of scripting, it is less troublesome to rely on a rendering engine that supports a rich set of widgets from the start.

## 4 Types and Constraints

The key elements in the description of a page are the type of the content data and the constraints. The type provides a concise representation of a set of values.

Constraints further restrict this set of values by stating dependencies and computation rules among parts of the values.

Types and constraints are quite general concepts and this section explains exactly which types and constraints our implementation supports. An early design decision was to base both on XML to exploit existing standards and tools as much as possible.

## 4.1 Types

The type structure underlying form-oriented analysis [6] comprises primitive types, entity (or record) types, and sequences thereof. Although it is simpler than UML's type structure and simpler than XML Schema, it is still sufficient to model form-based applications. To enable the reuse of tools (in particular validators) for XML Schema, we have designed a subset suitable for form-oriented analysis. This subset can be characterized as XML without attributes, choices, and complicated primitive types.

We include a number of primitive XML Schema types which have a straightforward mapping to common programming languages: `string`, `integer`, `decimal`, `date`, and `dateTime`. Restriction of primitive types is allowed because it amounts to adding constraints to a given type.

Complex types model the contents of XML elements. Each XML element (type) corresponds to the description of an entity type in form-oriented analysis. Complex types must not include attribute specifications. Furthermore, choices must not be used because form-oriented analysis models them using optional entities of multiplicity `0..1` and constraints. All other means of definition (including extension and restriction) are allowed.

## 4.2 Constraints

Constraints in the sense of form-oriented analysis [6] encompass aspects which are often considered as part of the type. An example for such a constraint is multiplicity. We step back from that view a little bit and leave constraints, which are usually part of an XML Schema type specification, in their usual place. That concerns multiplicities, range restrictions imposed on primitive types, length restrictions, regular expressions, and so on.

However, the constraints that XML Schema can impose are of a rather local nature. While such top-down dependencies are easy to specify, they are not sufficient for restrictions that relate different parts of a document. SPREANS covers two kinds of such constraints, co-occurrence constraints and evaluation constraints. In applications using standard browsers, the checking of these constraints is often left to the business logic. This practice leads to cluttered business logic and it causes latency to the user interface caused by the additional input-verification traffic between client and server. An often used alternative implements constraints using client-side scripting sometimes without ever stating the constraints formally. Because this approach is quite brittle there have been efforts to automatically generate the scripting code from a specification [3]. We



```

<!-- XML fragment -->
<bs:orderRegistration>
  <bs:userName></bs:userName>
  <bs:passwd></bs:passwd>
  <bs:repPasswd></bs:repPasswd>
</bs:orderRegistration>

<!-- Schematron schema -->
<?xml version="1.0"?>
<sch:schema version="ISO" xml:lang="en"
  xmlns:sch="http://www.ascc.net/xml/schematron">
  <sch:title>Registration Validation</sch:title>
  <sch:ns prefix="bs" uri="http://www.formcharts.org/bookshop"/>
  <sch:phase id="Minimal Check">
    <active pattern="pw-check"/>
  </sch:phase>
  <sch:pattern id="pw-check" name="Password Check">
    <sch:rule context="bs:orderRegistrationForm">
      <sch:assert test="bs:passwd = bs:repPasswd">
        The supplied passwords do not match.
      </sch:assert>
    </sch:rule>
  </sch:pattern>
</sch:schema>

```

Fig. 1. Repeated password check using a Schematron assertion

take the similar view that such tests are better specified in terms of constraints on the data model. The spreadsheet-based browser presents an opportunity to enforce such constraints without having to resort to scripting.

**Co-occurrence Constraints.** Many registration forms require a co-occurrence constraint. They contain one field for a user name and two fields for the user's password. In a valid submission the entries for the password fields must be non-empty and contain the same password. While XML Schema can prescribe non-emptiness, it cannot prescribe the equality of two fields.

SPREANS captures this kind of constraint using rules from Schematron [9], another Schema definition language for XML. Essentially, Schematron facilitates the specification of assertions and restrictions on XML documents by using XPath expressions. More specifically, Schematron provides two elements, **assert** and **report** (for restrictions), that both take an XPath expression as their **test** attribute. While **assert** requires that the **test** expression evaluates to true, **report** requires that it evaluates to false for passing the test. There is also a **pattern** element to group assertions and a **phase** element to group and reuse patterns. Figure 1 contains an example specification for validating the correctly repeated entry of a password in a registration form.

```

<bs:shoppingCart>
  <bs:cartItem>
    <bs:title>Quine: Word and Object</bs:title>
    <bs:price>12.46</bs:price>
    <bs:quantity>1</bs:quantity>
    <bs:posTotal/>
  </bs:cartItem>
  <bs:cartItem>
    <bs:title>Wittgenstein: Tractatus</bs:title>
    <bs:price>23.06</bs:price>
    <bs:quantity>1</bs:quantity>
    <bs:posTotal/>
  </bs:cartItem>
  <bs:totals>
    <bs:subtotal/>
    <bs:tax/>
    <bs:total/>
  </bs:totals>
  <bs:search></bs:search>
</bs:shoppingCart>

```

**Fig. 2.** Example: shopping cart

```

<fields>
  <field ref="/bs:shoppingCart/bs:cartItem/bs:title" />
  <field ref="/bs:shoppingCart/bs:cartItem/bs:price" />
  <field ref="/bs:shoppingCart/bs:cartItem/bs:quantity" />
  <field ref="/bs:shoppingCart/bs:cartItem/bs:posTotal"
    calculate="../bs:price * ../bs:quantity" />
  <field ref="/bs:shoppingCart/bs:totals/bs:subtotal"
    calculate="sum(..../bs:cartItem/bs:posTotal)" />
  <field ref="/bs:shoppingCart/bs:totals/bs:tax"
    calculate="../bs:subtotal * 0.16" />
  <field ref="/bs:shoppingCart/bs:totals/bs:total"
    calculate="../bs:subtotal + ../bs:tax" />
  <field ref="/bs:shoppingCart/bs:search" />
</fields>

```

**Fig. 3.** Shopping cart evaluation constraints

**Evaluation Constraints.** The next kind of constraints concerns dependencies where one value functionally depends on other values. Such constraints occur frequently in forms for business applications, for example, to compute the sum of the positions in an invoice or an order, to compute taxes, rebates, etc. Such evaluation constraints have been used before in constraint-based spreadsheet technologies [12, 1]. SPREANS specifies evaluation constraints with XPath because XPath provides the facilities to address the operands as well as the required operators and aggregate functions. In particular, SPREANS provides functionality

quite similar to that in the latest XForms proposal [7]. The evaluation constraints are SPREANS' incarnation of the formulas in a spreadsheet and their calculation proceeds in the well-known dependency-driven manner.

As an example, consider the content data of a shopping cart in Fig. 2. Here, the evaluation constraints should state that

- the **posTotal** element in each order item is computed from the **quantity** and the **price** elements,
- the **subtotal** element in the **totals** element contains the sum of all the **posTotal** elements,
- the **tax** element contains the tax rate applied to the **subtotal**, and
- the **total** element contains the sum of the **subtotal** and the **tax** elements.

The specification in Fig. 3 enforces exactly these constraints. It enumerates all input fields on a shopping cart page and specifies constraints on those that are calculated automatically.

Other features of the XForms proposal, like disabling fields or making parts of forms appear and disappear depending on entered values, are also supported by SPREANS.

## 5 Implementation

This section explains how our prototype implements the architecture outlined in Section 3. Figure 4 shows the overall processing of a response message in the browser. First, the browser retrieves all parts of the response message and parses them into DOM structures: the type of the content data, constraints on the content data, the initial content data of the page (*Client Page DOM*), a list of server action bindings, and the layout of the page (*View DOM*). The View DOM connects to the Client Page DOM via XPath expressions. The browser instantiates the layout with the data in the Client Page DOM by dereferencing

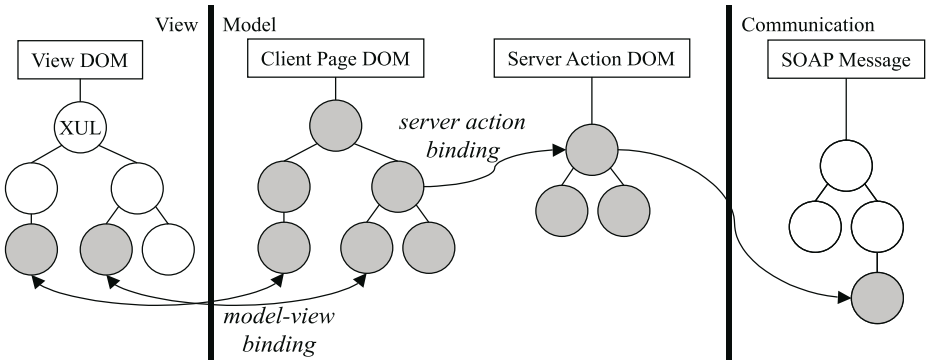
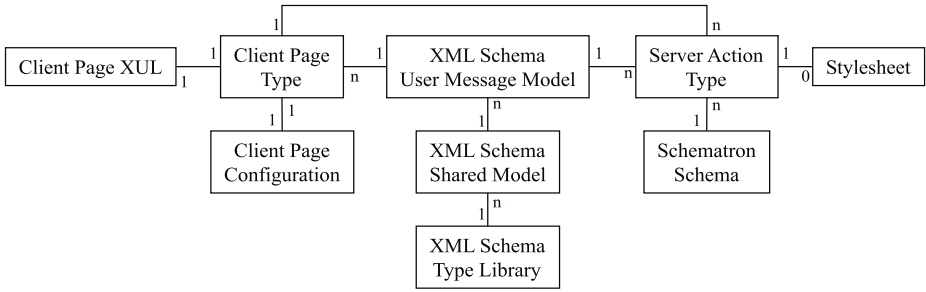


Fig. 4. Implementation of SPREANS



**Fig. 5.** Data organization in the implementation of SPREANS

the XPath pointers from the View DOM to the client page. After editing the page (which affects the View DOM and the Client Page DOM) the page is submitted to the server. At this point, the browser extracts a *Server Action DOM* from the Client Page DOM by applying an XSLT stylesheet to the content data. This stylesheet is part of the server action bindings. The resulting server action is validated with respect to the server action type and finally wrapped into a SOAP message and sent to the server.

Figure 5 shows a more detailed view of the data organization in the implementation. It differentiates the different layers of Schema definitions and it shows that each client page may trigger different server actions. Further, each server action is associated with a stylesheet for extracting the user message from the client page and several Schematron schemas for validation prior to submission. A client page consists of its configuration and its layout specification *Client Page XUL* (the View DOM of Fig. 4).

In principle, the SPREANS technology can be directly integrated into existing web browsers, either as a plugin or with JavaScript. We did not pursue that approach because we estimated that a direct implementation was cheaper and more robust.

## 6 Related Work

XAML (Extensible Application Markup Language) is an emerging technology for rich web clients. XAML augments and integrates features of the several client technologies found today on the web, i.e., HTML, SVG, PDF, WMF, and SWF. An essential part of the XAML effort targets the embellishment of web interfaces. Clearly, an embellished user interface can always be justified from a marketing point of view. While a rich web-client technology can bring a desktop metaphor to the web-application end user, the gain for users of enterprise applications is not clear because metaphors have to be learned just like the filling of a form.

There are proposals for a tight integration of office automation into an overall enterprise application system. For example, Whang et al[14] already proposed

to unify several typical office applications through the concept of *example element*, which comes along with a defined notion of processing by an integrated database management system. Nowadays, the XUL-based rich client technology RCPML (Rich Client Platform Markup Language) targets the integration of office automation into a surrounding system: the Lotus platform. Arie Segev [11] described the integration of applications into automatic business processes from an open, service-oriented viewpoint. Our approach is different in that we examine the spreadsheet as a single ubiquitous office automation application with respect to its capabilities as a general client.

Powerforms [3] is part of the web programming platform BIGWIG and defines a declarative language for expressing form input field formats and interdependencies between form input fields. Powerforms is similar to our approach for types and submission-time constraints. However, it does not support evaluation constraints. There are other approaches that allow for a type-safe web programming, for example, [2, 13, 5].

XForms [7] is the W3C-endorsed successor to HTML forms. XForms offers tag support for gathering and submitting structured data as XML documents. It is possible to specify constraints on data gathered by a form. The XForms approach does not only allow constraints on type correctness and required data entry, but allows to specify arbitrary validations and calculations with respect to data entered by the user. An event model defines hooks for handling violations of the specified constraints. There are some essential differences to our approach. First, XForms is a stand-alone proposal that supports a typed channel from the web browser back to the application. In contrast, our proposed browser is integrated into the fully typed framework of form-oriented analysis, where *all* messages between client and server are typed *including those from server to client*. Second, while XForms presents a step forward in the separation of content and layout, it still mingles control aspects in the view by either delivering the content and the layout in the same message or by having pointers to content instances in the layout message. Again, we take a step further by fully separating model and view aspects from each other and from control. The driving force of our browser is the metamodel, that is, the type of the message and constraints imposed on the instances of the type. Because of these reasons our browser easily integrates into service oriented architectures by considering it a consumer of web services provided by an enterprise application server. Third, XForms can only return entire subtrees of a model whereas our browser extracts the submitted data with an XSLT transformation. Fourth, while XForms grafts ideas like automatic calculation from the spreadsheet world underneath the functionality of a web browser, our preferred view is the other way round. We regard the spreadsheet as an accepted end-user technology and generalize it from the traditional grid view towards more expressive, embellished widget sets and with facilities to contact and render web services. With respect to embellished widget sets Forms/3 [4] is a well-known system with similar features, but a different goal: end-user programming.

## 7 Conclusion

We have presented a client technology for web-based services, SPREANS, that fosters from the outset a separation of content and layout. SPREANS extends a spreadsheet with browsing facilities. It offers strict typing, a constraint-based declarative programming model, and a default layout generation. It is based on XML and common patterns for XML schema design. Its firm foundation in spreadsheets does not have to influence the appearance. SPREANS allows the construction of arbitrary GUIs based on XUL, hence the usability is identical to an implementation of the same XUL interface with other technologies. With the combination of a spreadsheet client paradigm with web services access, SPREANS bridges the gap between web services and traditional web applications.

## References

1. Y. Adachi. Intellisheet: A Spreadsheet System Expanded by Including Constraint Solvers. In *HCC '01: Proc. IEEE Symposia on Human-Centric Computing Languages and Environments*, pages 173–179. IEEE Press, 2001.
2. D. Atkins, T. Ball, G. Bruns, and K. Cox. Mawl: a domain-specific language for form-based services. *IEEE Transactions on Software Engineering*, 25(3):334–346, 1999.
3. C. Brabrand, A. Møller, M. Ricky, and M. I. Schwartzbach. Powerforms: Declarative client-side form field validation. *World Wide Web Journal*, 3(4):205–214, 2000.
4. M. Burnett, J. Atwood, R. W. Djang, H. Gottfried, J. Reichwein, and S. Yang. Forms/3: A First-Order Visual Language to Explore the Boundaries of the Spreadsheet Paradigm. *J. of Functional Programming*, 11(2):155–206, Mar. 2001.
5. D. Draheim and G. Weber. Strongly Typed Server Pages. In *Proceedings of the 5th Workshop on Next Generation Information Technologies and Systems*, LNCS 2382. Springer, 2002.
6. D. Draheim and G. Weber. *Form-Oriented Analysis - A New Methodology to Model Form-Based Applications*. Springer, October 2004.
7. M. Dubinko, C. Software, L. L. Klotz, R. Merrick, and T. V. Raman. XForms 1.0 – W3C Recommendation. Technical Report REC-xforms-20031014, World Wide Web Consortium, October 2003.
8. G. Fischer, E. Giaccardi, Y. Ye, A. Sutcliffe, and N. Mehandjiev. Meta-Design: a Manifesto for End-User Development. *Communications of the ACM*, 47(9):33–37, November 2004.
9. I. O. for Standardization. Final Committee Draft of ISO Schematron. Technical Report ISO/IEC FDIS 19757-3, ISO/IEC.
10. C. Jones. End-User Programming. *IEEE Computer*, 28(9):68–70, 1995.
11. A. Segev. Enabling Design-Centric eBusiness Applications. In *NGITS'02: 5th International Workshop on Next Generation Technologies and Systems*, LNCS 2382. Springer, 2002.
12. M. Stadelmann. A spreadsheet based on constraints. In *UIST '93: Proc. 6th Annual ACM Symposium on User Interface Software and Technology*, pages 217–224, New York, NY, USA, 1993. ACM Press.

13. P. Thiemann. Wash/CGI: Server-side Web Scripting with Sessions and Typed, Compositional Forms. In *Practical Aspects of Declarative Languages (PADL'02)*, January 2002.
14. K.-Y. Whang, A. Ammann, A. Bolmarcich, M. Hanrahan, G. Hochgesang, K.-T. Huang, A. Khorasani, R. Krishnamurthy, G. Sockut, P. Sweeney, V. Waddle, and M. Zloof. Office-by-example: an integrated office system and database manager. *ACM Trans. Inf. Syst.*, 5(4):393–427, 1987.

# Dynamic Construction of User Defined Virtual Cubes<sup>\*</sup>

Dehui Zhang<sup>1</sup>, Shaohua Tan<sup>1</sup>, Dongqing Yang<sup>2</sup>, Shiwei Tang<sup>1,2</sup>,  
Xiuli Ma<sup>1</sup>, and Lizheng Jiang<sup>2</sup>

<sup>1</sup> National Laboratory on Machine Perception, School of Electronics Engineering  
and Computer Science, Peking University, Beijing 100871, China

<sup>2</sup> School of Electronics Engineering and Computer Science, Peking University,  
Beijing 100871, China

{dhzhang, tan, maxl, lzjiang}@cis.pku.edu.cn  
{dqyang, tsw}@pku.edu.cn

**Abstract.** OLAP provides an efficient way for business data analysis. However, most up-to-date OLAP tools often make the analysts lost in the sea of data while the analysts usually focus their interest on a subset of the whole dataset. Unfortunately, OLAP operators are usually not capsulated within the subset. What's more, the users' interests often arise in an impromptu way after the user getting some information from the data. In this paper, we give the definition of users' interests and propose the user-defined virtual cubes to solve this problem. At the same time, we present an algorithm to answer the queries upon virtual cube. All the OLAP operators will be encapsulated within this virtual cube without superfluous information retrieved. Experiments show the effectiveness and efficiency of the virtual cube mechanism.

## 1 Introduction

OLAP has been an important tool for enterprise data analysis, and as a method of organizing analysis oriented data, multi-dimensional data cubes enable users to surf data conveniently. Meanwhile, pre-computed aggregations enable users to drill down and roll-up data views freely only with a little response time.

Since E.F.Codd proposed the concept OLAP in 1993 [4], the related research can roughly be divided into two phases. In the first phase, people focused their eyes on architectures of OLAP system, multi-dimensional data model and OLAP operators' implementation. In the second phase, in order to make the OLAP tools be more automatic and intelligent, researchers have made a lot of effort. Some advanced operators were proposed, including intelligent roll-up [14] and drill-down [15], and user-adaptive exploration [13]. Semantic summarization [9] [10], iceberg cubes [12] and dimension grouping [7] improve the effectiveness of OLAP in different perspectives. Unfortunately, there is still another pending problem with state-of-the-art OLAP techniques as below:

When an OLAP user drills down (or rolls up), from the dimension member "China" to the next level "city", all the cities of China will be fetched. However, in

---

<sup>\*</sup> This work is supported by the National Natural Science Foundation of China under Grant No.60473072.



real application, users often want to focus their analysis only on a sub-dataset. For example, a director of Walmart may be interested only in sales about China, Japan, Korea and Singapore. At the same time, if they roll up from “city” to “country”, only the above four countries will be valuable to him. In short, on one hand, the users want to perform analysis within a subset, and on the other hand, they want to view this subset as the whole cube, within which all the OLAP operators should be encapsulated.

With current techniques, in order to achieve this goal, a user must take a series of actions: selecting the appropriate cuboid, performing SLICE or DICE, and summing up the sales of these four countries. Apparently, this takes more time to get the wanted information, because too many steps should be followed. Since this kind of impromptu user demand may occurs frequently, a mechanism is needed.

Updating the dimension structures [2] of dataset can reach this goal. However, the operators for modifying the dimension schema and dimension instance cost too much for data reorganizing or updating. On the other hand, updating to the dimension structure cannot solve users’ impromptu interest, because ever-changing dimension schema is unimaginable to data cubes.

A constraint cube is defined in [8]. The author first materializes a constraint cube, and then performs singularity discovery mining task. But complete materialization of such cubes is infeasible, because too much storage is needed. In this paper, we propose the user-defined virtual cubes to solve this problem. First, user declares his interest of the data, i.e. tells the system in which part of the original data he is interested. The user interest forms a filter against the base dataset, and results in a subset. Just like that there is a data cube from original dataset, all the possible aggregations of this subset form a data cube too, except that it does not exist physically. So, we call this cube a user defined virtual cube.

In this paper, we propose a greedy algorithm to select the cuboids of the virtual cube to materialize based on a cost model. The cost models for materialized views selection are presented in [6] [1] [11]. These models are based on disk accessing and views maintenance cost. Such models are usually based on the cost of answering a given set of queries using materialized views set. We consider this problem from another perspective that we choose the most profitable cuboid to materialize. So, we make some modifications on these models, and we use OLAP operators as a cost unit to weight the profit of a cuboid.

Since users’ interests usually arise in an impromptu way, we cannot construct all the possible virtual cubes from the word go. Instead, we must construct them dynamically along with OLAP analysis process. In this paper, we propose a greedy algorithm to materialize the views of virtual cube. We develop a cost model to weigh which cuboid is the most profitable. At the same time, we present an algorithm to answer the queries upon virtual cube. All the OLAP operators will be encapsulated, and no superfluous information retrieved.

The remainder of the paper is organized as follows. The problem will be formulated in section 2. In Section 3, we will propose a greedy algorithm for dynamic construction of virtual cube and a cost model on which the algorithm is based. The algorithm to answer queries over virtual cube will be described in section 4, and in section 5, we present experiment result of this algorithm. At last, conclusions and future works are presented in section 6.

## 2 Problem Formulation

### 2.1 Multi-dimensional Data Model

In previous work, Hurtado et al. [5] introduced a multidimensional model. Due to space limitations we will only give a quick overview of that model, and refer the reader to related works for details.

**Definition 2.1**[5]. A dimension schema is a directed acyclic graph  $G(L, \equiv)$  where each node represents a dimension level (or an attribute), and  $\equiv$  is an aggregation relation between levels such that its transitive and reflexive closure,  $\equiv^*$ , is a partial order with a unique bottom level  $l_{inf}$ , and a unique top level  $All$ .

**Definition 2.2**[5]. An *instance* of a dimension is an assignment of a set of *elements* to each dimension level. Moreover, between every pair of levels  $l_i$  and  $l_j$  such that  $l_i \equiv l_j$  there is a function  $\rho_{l_i}^{l_j}$  called rollup. A dimension instance must satisfy the *consistency condition*, meaning that for every pair of paths from  $l_i$  to  $l_j$  in  $\equiv$ , the composition of the rollup functions yields identical functions. The number of distinct *element* of a level is called its cardinality.

In real application, a dimension's structure can be as complicated as that it has several aggregation paths, while we can view each path as a separate structure. For the convenience of narration, we suppose that each dimension has only one aggregation path in this paper.

**Definition 2.3.** A dimension  $D$  can be shown as  $D = (H_1, H_2, \dots, H_n)$ , where  $H_i$  is a dimension level, and  $H_i \equiv H_j$ , for every pair of  $i, j$  that  $i \geq j$ . The set  $R_D = \{(H_1, H_2, \dots, H_i) | 0 \leq i \leq n\}$  is called roll-up set of  $D$ , and  $|R_D| = n+1$ . When  $i=0$ , then  $R_D = \{All\}$ , see *figure-1*.

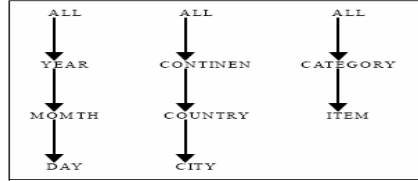
**Definition 2.4.** A multi-dimensional dataset  $R$  can be represented as  $R(D_1, D_2, \dots, D_n; M)$ , where  $D_i$  is the  $i^{\text{th}}$  dimension of  $R$ , and  $M$  is the measure attribute.  $R$  is the schema of the dataset, and a dataset with schema  $R$  is called an *instance* of  $R$ .

In this paper, we use  $R$  to represent both the schema and the instance if no confusion is incurred.

**Definition 2.5.** A *data cube*  $C$  from  $R$  is the set of all the possible aggregated dataset of  $R$ , and  $R$  is called *base dataset*. Each possible aggregated dataset is called a *cuboid*, represented as  $\text{cuboid}(N)$ , where  $N$  is the set of dimension levels,  $N \in 2^{\{A_1, A_2, \dots, A_n\}}$ , and  $A_i \in R_{D_i}$  for the  $i^{\text{th}}$  dimension of  $R$ , see *figure-2*.

**Definition 2.6.** All the schemas of possible aggregations of  $R$  form an algebra lattice. A lattice  $L$  of a data cube  $C$  is a partially ordered set in which all finite subsets have a least upper bound and a greatest lower bound. It is represented as  $L: (S, \leq)$ , where  $S$  is the set of cuboids and  $\leq$  is the partial order of aggregation on  $S$ , and for each pair  $x, y$ ,  $x \leq y$ ,  $y$  is one of  $x$ 's parents, and  $x \leq x$ .

**Example 1.**  $R(\text{time, region, product; sales})$  is a sales multi-dimensional dataset with three dimensions(time, region and product) and one measure(sales). Each dimension has its own hierarchies as: time=(All, year, month, day), region=(All, continent, country, city) and product=(All, category, item).



**Fig. 1.** Aggregation path of dimension

## 2.2 User Interest

When performing data analysis, the users usually have their own expectations to the data. Even though they have no knowledge about the data, they would like to find something interesting within a given subset of the data, after getting familiar to the data. Different users will put their eyes over different data, while the data they pay less attention may become the superfluous information, even noise.

In this paper, we call the sub-dataset where a user is interested his *interest*. User interest can be expressed with a filter against the original base dataset, with a sub-dataset resulted.

**Definition 2.7.** A user interest  $I$  of  $R$  is a predicate over dimension levels.  $I = p_1 \vee p_2 \vee \dots \vee p_k$ , where  $p_i = f_{(i,1)} \wedge f_{(i,2)} \wedge \dots \wedge f_{(i,k)}$ ,  $k$  is the number of attributes of  $R$  and  $f_{(i,j)}$  is a predicate about  $j^{\text{th}}$  attribute of  $R$ . Let  $R' = \{t \mid t \in R \text{ and } t \text{ meets } I\}$  and  $I_D = \{A_j \mid A_j \text{ is an attribute, and } \exists p_i = f_{(i,1)} \wedge f_{(i,2)} \wedge \dots \wedge f_{(i,k)}, \text{ where } f_{(i,j)} \text{ is not a tautology}\}$ , we called  $I_D$  *interesting attributes set*, and  $A_j$  is an interesting attribute.

**Definition 2.8.** Suppose that the total number of the dimension elements be  $\alpha$ , and there only  $\beta$  elements meet the user interest  $I$ . Let  $\delta = \beta / \alpha$ , we call  $\delta$  the *filter factor* of  $I$ .

**Example 2.** Let a user's interest  $I$  is "the sales during SARS period" over time dimension, and  $I$  can be formulated as  $I = \text{"time is between (2003-01-14, 2003-06-29)"}$ , and  $I_D = \{\text{year, month, day}\}$ .

**Example 3.** Another user interest "the pan-pacific sales during SARS period" can be formulated as  $I = \text{"time is between (2003-01-14, 2003-06-29)" } \wedge \text{"region is belong to pan-pacific district"}$ , and  $I_D = \{\text{year, month, day, continent, country}\}$  accordingly.

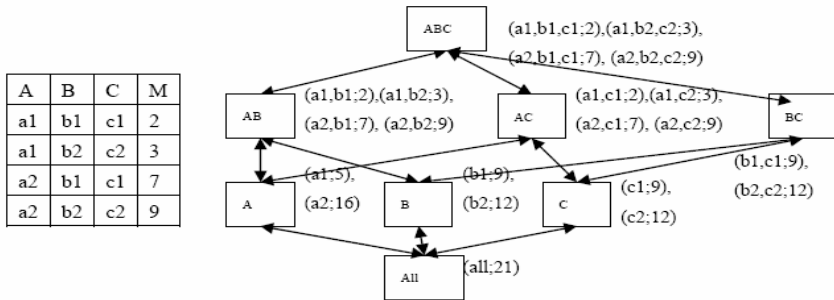
## 2.3 User Defined Virtual Cube

A user interest results in a subset of the base dataset, but it is different when applied to the data cube.

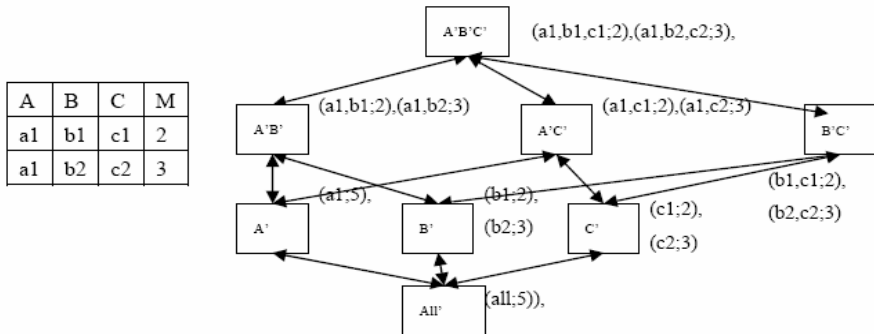
**Definition 2.9.** Let  $C$  be the data cube from base dataset  $R$ , the user interest  $I$ , and  $R'=\{t| t \in R \text{ and } t \text{ meets } I\}$ . A user defined *virtual cube* is the set of all the possible aggregated datasets of  $R'$ , represented as  $C'=I(C)$ . Each cuboid  $c \in C'$  is called a *virtual cuboid* and  $C$  is called the *original cube*.

Since user interest does not change the schema of base dataset,  $C$  and  $C'$  will share the same cuboid schema. For each pair of cuboids  $\langle R, S \rangle$ ,  $R \in C$   $S \in C'$ , if  $R, S$  are sharing the same schema, we call  $\langle R, S \rangle$  a *dataset pair*. In this paper, we use a function *pair*( $S$ ) to represent  $S$ 's dataset pair of original data cube  $C$ , that is to say for a dataset pair  $\langle R, S \rangle$ ,  $R \in C$  and  $S \in C'$ ,  $R = \text{pair}(S)$ .

**Example 4.** Suppose a dataset  $R$  has three dimensions:  $A, B$  and  $C$ , and it has one measure  $M$ . The figure-2 shows the dataset and its cube lattice. Let the user interest  $I$  be  $A='a1'$ , the figure-3 shows the virtual cube lattice and all the cells of this virtual cube. The cuboid pair  $\langle (AB), (A'B') \rangle$  is a dataset pair, where the cuboid  $(AB)$  is shown in the figure-2 and the cuboid  $(A'B')$  is shown in the figure-3.



**Fig. 2.** The lattice of the original cube



**Fig. 3.** The lattice of the virtual cube

### 3 Virtual Cube Construction

After a user declares his interest, in order to accelerate access to his favorite data, we should construct the virtual cube accordingly. The construction process is dynamic, for it does not exist when the initial cube is deployed. It is constructed in an impromptu style on analyzer's demand—declaring their interests to the data.

In this paper, we want to find a proper way to construct the virtual cube and process queries over it. Firstly we take a glance at three naïve algorithms. Then, we propose a storage constraint greedy algorithm to select cuboids to materialize based on a cost model.

#### 3.1 Naïve Algorithms

There will be three naïve materialization strategies: fully materialized, none materialized and root materialized. Since the ideas are easy and straightforward, we will make no additional explanation on them, and we will give the analysis of them.

*Fully Materialized Algorithm (FMA)* provides least response delay for accessing virtual cubes, and there is a substantial cube algorithm to be chosen from [3]. But computing all the possible aggregations takes much I/O and CPU time and large volume of storage.

*None Materialized Algorithm (NMA)* is the best annotation for impromptu virtual cubes. Under the circumstances, all the queries against the virtual cube must be mapped to queries against original cubes. This means that the filter representing user interest will be run as many times as queries. If queries over this virtual cube are frequent, the response time may be unacceptable.

The base dataset of a virtual cube will be materialized in the *Root Materialized Algorithm (RMA)*. If RMA is adopted, less storage is consumed than FMA, and less delay is incurred than NMA. However, probably it can neither take full advantage of storage nor reach the best accessing performance.

For a given data cube with  $n$  cuboids or datasets, and  $\Omega$  available storage constraint, the cost of finding the best materialization combination is  $O(2^n)$ . When there are many dimensions or levels of a dimension, an algorithm with such cost is unfeasible, so we must find an applicable substitute. Next, we will present our greedy algorithm and the cost model on which it is based.

#### 3.2 Greedy Algorithm

##### 3.2.1 Cost Model

When a user performs OLAP analysis, the most often used operators are *SELECT* and *AGGREGATION*. For completely materialized data cube, OLAP analysis is just to choose right dataset to run *SELECT* and show the result on the screen. On the other hand, for partly materialized data cube, if the requested dataset is not materialized, the OLAP system must first choose appropriate dataset to run *AGGREGATION*, then run *SELECT* on the intermediate result to get the final result.

In our cost model, we just regard *SELECT* and *AGGREGATION* as two cost unit. They are noted as  $C_{sel}(N)$  and  $C_{agg}(N, O)$  respectively, where  $N$  is the size of input dataset and  $O$  is the size of aggregation output. There are many papers addressing cost

of these two operators. For *SELECT*,  $O(1) \leq C_{sel}(N) \leq O(N)$ , and for *AGGREGATION*, the most effective algorithms is  $O(N)$  [16].

In this paper, we use how much reduction a materialized dataset can provide to describe its profit. In fact, this cost model is a profit model.

**Definition 3.1.** For a given data cube  $C$ , user interest  $I$ ,  $C' = I(C)$ , let  $L(S, \leq)$  and  $L'(S', \leq)$  are the lattices of  $C$  and  $C'$ . For each cuboid  $c'(N') \in C'$ ,  $c(N) \in C$ ,  $c = \text{pair}(c')$ , and  $b \in P_c(c') = \{t | t \in S, c \leq t\}$ , we call  $b$  as *virtual parents* of  $c'$ , and if  $b \in IP_v(c') = \{t(M) | t \in S, c \leq t, I_D \subseteq M\}$ , we call  $b$  as *virtual interesting parents* of  $c'$ , the member  $f$  of  $IP_v(c')$  whose size is the minimum is called *virtual father* of  $c'$ , representing by  $vf_{c'}$ .

Our algorithm requires knowledge of the sizes of each cuboid of  $C$ . There are several ways of estimating the sizes of cuboids. One commonly used approach is to run aggregation algorithm on a statistically significant but small subset of base dataset. This subset can be obtained by sampling the base dataset [6].

**Definition 3.2.** For each cuboid  $c'(N') \in C'$ ,  $c(N) \in C$ ,  $c = \text{pair}(c')$ , and if  $I_D \subseteq N$ ,  $|c'| = \delta |c|$ , else  $|c'| = |c|$ .

**Definition 3.3.** The *profit* of materialization of a virtual cuboid  $c'(N)$  can be defined as following:

- 1) If  $I_D \subseteq N$ ,  $\text{profit}(c') = 0$ ;
- 2) Else,  $\text{profit}(c') = \sum_{t \in c'} [C_{sel}(|vf_t|) + C_{agg}(\delta |vf_t|, |t|) - C_{agg}(|c'|, |t|)]$ .

Here is the explanation of this formula: if  $c'$  is materialized, for each cuboid  $m$  that  $m \leq c'$ ,  $m$  can be computed from  $c'$  with only one aggregation operator. Otherwise the virtual father  $f$  of each cuboid  $t$  must be found, and first impose select operation on  $f$ , then aggregate the temporary result to  $t$ .

### 3.2.2 The Greedy Algorithm

As a user of the OLAP system, we should not suppose that he have unlimited available storage space. In fact, usually there are many users of the OLAP system, and different users have different levels of priority. In this paper, we suppose that a user construct his virtual cubes under the constraint of totally  $\Omega$  available storage space.

We first compute profit of each cuboid of the virtual cube, and then sort the cuboids with their profits. Each time we choose the most profit cuboid satisfying the remaining storage space to materialize, see *algorithm-1*.

*Algorithm-1:* constructing a virtual cube

*Input:*  $C, I, \Omega$

*Output:*  $L', L' \subseteq C'$

1:  $L' = \Phi$ ;

2: evaluate the size of cuboids;

3: compute filter factor  $\delta$  of  $I$ ;

4: compute profit of each cuboid of  $C'$ ;

5: sort cuboids of  $C'$  with their profits;

```

6: while  $\Omega > 0$ 
    Select the current most profitable cuboid  $t$  from  $C'$ ;
     $L' = L' \cup \{t\}$ ;  $\Omega = \Omega - |t|$ ;  $C' = C' - \{t\}$ ;
7: materialize each cuboid  $t$  if  $t \in L'$ ;

```

Here we give the analysis of this algorithm. Both the 1<sup>st</sup> and 3<sup>rd</sup> step are of  $O(1)$  cost. Suppose that there be  $w$  cuboids, the 2<sup>nd</sup>, 4<sup>th</sup> and 6<sup>th</sup> steps are all of  $O(w)$  cost, and for most application cube this cost can be viewed as a constant. The 5<sup>th</sup> step is of  $O(w \log w)$ . While to materialize the cuboids, we can make a little modification on the BUC[3] algorithm, just simply noting the selected cuboids. So the cost of 7<sup>th</sup> step will be  $O(\eta \log \eta)$ , where  $\eta$  is the number of tuples in the  $R'$ .

## 4 Query over Virtual Cube

After virtual cube is constructed, we now can answer queries over it using materialized views. A naïve idea is that, for each requested cuboid of virtual cube, we should first found its materialized parents in this virtual cube. However, this may not be the best strategy, for the requested cuboid may be computed from its materialized parents of original cube, which possibly cost less than from its parents.

In this paper, we present an optimized query re-writing algorithm to answer queries over virtual cube.

**Definition 4.1.** An OLAP *query* usually has the following format  $Q(\text{measure}, \text{cuboid}, \text{condition})$ , where *measure* is aggregated value of a cell, *cuboid* is representing requested dataset, and *condition* is restrictive predict over this dataset with filter factor  $\varepsilon$ . Since query condition is usually specification of a level member,  $\varepsilon$  is easy to estimate with the reciprocal of this level's cardinality.

*Algorithm-2:* query answering with virtual cube

*Input:*  $C, I, C', Q(m, b, c)$

*Output:*  $Q'$

```

1: compute filter factor  $\varepsilon$  of  $c$ ;
2: compute filter factor  $\bullet$  of  $I$ ;
3: for all materialized  $s \in C'$  and  $b \leq s$ 
    compute  $x = \min(C_{agg}(|s|, |b|) + C_{sel}(|b|))$ ;
4: for all  $t \in IP_v(b)$ 
    compute  $y = \min(C_{agg}(\delta \varepsilon |t|, |b|) + C_{sel}(|t|))$ 
5: if  $I_b$  is subset of the schema of  $b$ ,
     $Q' = (m, \text{SELECT}(\text{pair}(b), I \wedge c), \text{TRUE})$ ;
6: else, if  $(x < y)$ , let  $C_{agg}(|u|, |b|) + C_{sel}(|b|) = x$ ,
     $Q' = (m, \text{AGGREGATION}(\text{SELECT}(u, c), b), \text{TRUE})$ ;
7: else, let  $C_{agg}(\delta \varepsilon |v|, |b|) + C_{sel}(|v|) = y$ ,
     $Q' = (m, \text{AGGREGATION}(\text{SELECT}(v, I \wedge c), b), \text{TRUE})$ ;

```

Here we give the analysis of this algorithm. Every step of this algorithm is  $O(1)$  cost except for the 3<sup>rd</sup> and the 4<sup>th</sup> step. In fact, if there are  $w$  cuboids, these two steps will be  $O(w)$  cost.

**Theorem-1.** For each query over virtual cube, let  $x$  be the cost of the query answering with virtual cube, and let  $y$  be the cost with original cube, then  $x \leq y$ .

**Proof.** If otherwise  $x > y$ , according to the 7<sup>th</sup> step of algorithm-2, this query will be answered with original cube. This is a conflict, so  $x$  is not greater than  $y$ . That is to say, for each query over virtual cube, no delay is incurred by virtual cube. ■

OLAP queries over data cube, such as drill-down, are often used to select data from a cuboid with specified condition, and this condition usually is as simple as that only one level member is specified. In the algorithm-2, besides making best of materialized virtual cuboids, we also try to push the condition down to dataset before an aggregation operator is invoked.

## 5 Experiments

In this paper, we design the experiments based on the dataset R (year, month, day; continent, country, city). We adopt the user interest in example-2. The cardinalities of dimension levels are represented as a vector (12, 144, 4383, 5, 150, 10000).

### 5.1 Preliminary Hypothesis

It is always the case that, OLAP queries can be divided into two categories. One of them is to request an entire cuboid, and the other is to request the cells of a cuboid with specified attributes members. The costs of answering these two kinds of queries are of a little difference. The queries of the former category need to scan the whole cuboid. The cost of answering the latter kind of queries depends on the indexing strategies. If the requested attribute is indexed, the cost will be  $O(1)$ , otherwise, scanning the cuboid is unavoidable.

In our experiments, we take only the former kinds of queries into consideration. The improvements brought by virtual cube are to reduce the running times of  $I$ , so this hypothesis has no effect on our conclusions.

### 5.2 Experiment Results

In this paper, we suppose that the queries distribute uniformly over cuboids. In fact, every query over virtual cube will have no more response time if it is answered using the algorithm-2 at least it is equal to the response that it is answered using original cube. So, the distribution of queries makes little difference.

Since a single query is not enough to show the performance contrast of using virtual cube and the traditional techniques, we pose one query on each cuboid, and we use the average response time as the benchmark of performance.

#### 5.2.1 Time Cost vs. Size of Base Dataset with Fixed Storage Constraint

In this experiment, we use  $M=512\text{MB}$  as available storage, and the base datasets have  $256 \times 10^4$ ,  $512 \times 10^4$ ,  $1024 \times 10^4$ ,  $1536 \times 10^4$ ,  $2048 \times 10^4$ ,  $3072 \times 10^4$ , and  $4096 \times 10^4$  tuples respectively.

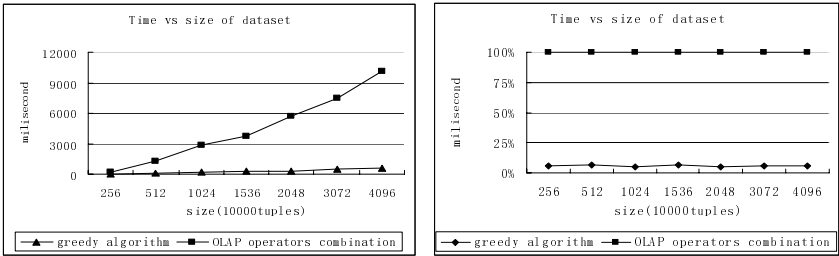


The sizes of cuboids for each dataset are listed in table-1.

**Table 1.** Total cells in original and virtual cubes

| Size( $10^4$ tuples) | Cube cells | Vcube cells | Ratio  |
|----------------------|------------|-------------|--------|
| 256                  | 2694543    | 744519      | 27.63% |
| 512                  | 5389088    | 1489184     | 27.63% |
| 1024                 | 10778175   | 2980165     | 27.65% |
| 1536                 | 16167267   | 4457316     | 27.57% |
| 2048                 | 21556353   | 5630519     | 26.12% |
| 3072                 | 32334535   | 9286478     | 28.27% |
| 4096                 | 43112710   | 11899108    | 27.60% |

The experiment results are as following:



**Fig. 4.** Time versus size of dataset with fixed available storage

For this experiment, the query response time of original cube is nearly 16 times of that of virtual cube. All the virtual cuboids are materialized. The total cells of virtual cube are one-fourth to that of original cube. From the figures above, we know the query response time ratio is nearly stable regardless of the size of datasets.

In fact, under our query distribution hypothesis, to answer the queries needs to scan the virtual cube once. However, for answering the queries with original cube, each query requesting a virtual cuboid should be answered with its virtual father. So, for this dataset, in order to answer the queries, original cuboid (all; continent, country, city) need to be scanned four times respectively. The size summation of these four cuboids is four times to the size of virtual cube. So the ratio is 16.

From the experiment results and theorem-1, we can safely infer that there will be a linear relationship between the response times only that the ratio will be different.

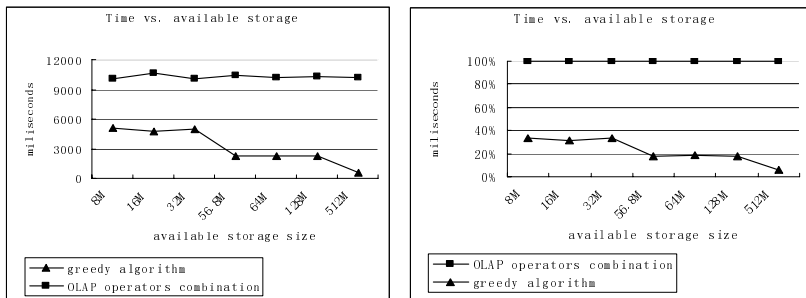
**5.2.2 Time Cost vs. Storage Constraint with Fixed Size of Base Dataset**

In this experiment, we use  $4096 \times 10^4$  tuples as base dataset, and the available storage spaces are 512MB, 128MB, 64MB, 56.8MB, 32MB, 16MB and 8MB respectively. The materialization list is shown in table-2.

**Table 2.** Materialization list

| Storage | Materialized cuboids  |
|---------|---|
| 512MB   | All cuboids   |
| 128MB   | Except (year, month, day; continent, country, city)   |
| 64MB    | Except (year, month, day; continent, country, city)   |
| 56.8MB  | (year, month; continent, country, city) (year; continent, country, city)<br>(all; continent, country, city) (all, all)  |
| 32MB    | Except (year, month, day; continent, country, city)<br>(year, month, day; continent, country) (year, month, day; continent)<br>(year, month, day; all)and (year, month; continent, country, city) |
| 16MB    | The same as that when 32MB is available   |
| 8MB     | The same as that when 32MB is available   |

The experiment results are as following:



**Fig. 5.** Time versus available storage with fixed size of dataset

For this experiment, the above figures apparently indicate that, less storage used more response time cost. The best performance is reached when all the virtual cuboids are materialized. When the available storage is 8MB, 16MB or 32MB, the materialization strategies of virtual cube are same. So does when it is 64MB and 128MB. This result provides farther evidence to what we inferred from the experiment depicted in 5.2.1.

## 6 Conclusions and Future Works

In this paper, we studied the problem that answering queries over user interested subset of data cube. We proposed dynamically constructed virtual cube to solve this problem. A cost model is employed in this paper to evaluate the profit of the materialization of a virtual cuboid. In order to decide which virtual cuboid should be materialized, a greedy algorithm is developed base on our cost model. We also

presented an algorithm to answer the queries over virtual cubes. Evaluations and experiments showed the effectiveness and efficiency of the virtual cube mechanism.

However, to construct a virtual cube for every user's interest is not the best solution to this problem. In the future, how to share virtual cubes among users and how to maintain the virtual cubes will be our next effort.

**Acknowledgements.** Thanks to Prof. Jiawei Han for his advice when we were preparing this work.

## References

1. Agarwal, S, Agrawal, R., Deshpande, P.M., et al, "On the Computation of Multi-dimensional Aggregations", In Proc. of the 22nd VLDB Conference, India, 1996.
2. Alejandro A. Vaisman, Alberto O. Mendelzon. Walter Ruaro, and Sergio G. Cymerman. Supporting Dimension Updates in an OLAP Server. Proceedings of the 7th ACM international workshop on Data warehousing and OLAP, November 12-13, 2004, Washington, DC, USA.
3. K. Beyer and R. Ramakrishnan. Bottom-up computation of sparse and iceberg cubes. In Proc. 1999 ACM-SIGMOD Int.Conf. Management of Data (SIGMOD'99), pages 359–370, Philadelphia, PA, June 1999.
4. E. F. Codd, S. B. Codd, and C. T. Salley. Beyond decision support. Computer World, 27, July 1993.
5. Carlos A. Hurtado, Alberto O. Mendelzon, Alejandro A. Vaisman. Updating OLAP dimensions. Proceedings of ACM DOLAP'99, 1999.
6. V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing data cubes efficiently. In Proc. 1996 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'96), pages 205–216, Montreal, Canada, June 1996.
7. Xiaolei Li, Jiawei Han, Hector Gonzalez. High-Dimensional OLAP: A Minimal Cubing Approach. VLDB, 2004.
8. LI Cui-Ping, LI Sheng-En, WANG Shan, DU Xiao-Yong. A Constraint-Based Multi-Dimensional Data Exception Mining Approach. Journal of Software. 2003.
9. Lakshmanan VS, Pei J, Han JW. Quotient cube: How to summarize the semantics of a data cube. In: Proc. of the 28th Int'l Conf. on Very Large Data Bases. Hong Kong: Morgan Kaufmann Publishers, 2002. 778789.
10. L.V.S. Lashmanan, J. Pei, and Y. Zhao. QC-Trees: An efficient summary structure for semantic OLAP. In Proc. 2003 ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'03), San Diego, California, June 2003.
11. Rada Chirkova, Chen Li. Materializing Views with Minimal Size To Answer Queries. PODS 2003.
12. Raymond T. Ng, Alan S. Wagner, and Yu Yin. Iceberg-cube computation with PC clusters. In Proc. 2001 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'01), Santa Barbara, CA, May 2001.
13. Sarawagi S. User-adaptive exploration of multidimensional data. In: Proc. of the 26th Int'l Conf. on Very Large Data Bases. Cairo: Morgan Kaufmann Publishers, 2000. 307–316.
14. Sathe G, Sarawagi S. Intelligent rollups in multidimensional OLAP data. In: Proc. of the 27th Int'l Conf. on Very Large Data Bases. Roma: Morgan Kaufmann Publishers, 2001.531–450.

15. Sarawagi S. Explaining differences in multidimensional aggregates. In: Proc. of the 25th Int'l Conf. on Very Large Data Bases. Edinburgh: Morgan Kaufmann Publishers, 1999.
16. Young-Koo Lee, Kyu-Young Whang, Yang-Sae Moon, and Il-Yeol Song. A One-Pass Aggregation Algorithm with the Optimal Buffer Size in Multidimensional OLAP. VLDB 2002.

# Automatic Discovery of Regular Expression Patterns Representing Negated Findings in Medical Narrative Reports

Roni Romano<sup>1</sup>, Lior Rokach<sup>2</sup>, and Oded Maimon<sup>1</sup>

<sup>1</sup> Department of Industrial Engineering, Tel Aviv University,  
Ramat Aviv, Israel  
ronir@eng.tau.ac.il, maimon@eng.tau.ac.il

<sup>2</sup> Department of Information Systems Engineering, Ben-Gurion University of the Negev,  
Beer-Sheva, Israel  
liorrk@bgu.ac.il

**Abstract.** Substantial medical data such as discharge summaries and operative reports are stored in textual form. Databases containing free-text clinical narratives reports often need to be retrieved to find relevant information for clinical and research purposes. Terms that appear in these documents tend to appear in different contexts. The context of negation, a negative findings, is of special importance, since many of the most frequently described findings are those denied by the patient or subsequently “ruled out.” Hence, when searching free-text narratives for patients with a certain medical condition, if negation is not taken into account, many of the documents retrieved will be irrelevant. In this paper we examine the applicability of a new pattern learning method for automatic identification of negative context in clinical narratives reports. We compare the new algorithm to previous methods proposed for the same task of similar medical narratives and show its advantages. The new algorithm can be applied also to further context identification and information extraction tasks.

**Keywords:** Medical Informatics, Text Classification, Machine Learning, Information Retrieval.

## 1 Introduction

Medical narratives present some unique problems. When a physician writes an encounter note, a highly telegraphic form of language may be used. There are often very few (if any) grammatically correct sentences, and acronyms and abbreviations are frequently used. Very few of these abbreviations and acronyms can be found in a dictionary and they are highly idiosyncratic to the domain and local practice. Often misspellings, errors in phraseology, and transcription errors are found in dictated reports.

Researchers in medical informatics suggested methods for automatically extracting information contained in narrative reports for decision support [6], guideline implementation [10], and detection and management of epidemics [12].

Nevertheless most of the researches have concentrates on methods for improving information retrieval from narrative reports [11, 17, 20]. A search for patients with a

specific symptom or set of findings might result in numerous records retrieved. The mere presence of a search term in the text, however, does not imply that records retrieved are indeed relevant to the query. Depending upon the various contexts that a term might have, only a small portion of the retrieved records may actually be relevant.

A number of investigators have tried to cope with the problem of a negative context. NegExpander [1] uses syntactic methods to identify negation in order to classify radiology (mammography) reports. While NegExpander is simple in that it recognizes a limited set of negating phrases, it does carry out expansion of concept-lists negated by a single negating phrase.

MedLEE [9] performs sophisticated concept extraction in the radiology domain. The MedLEE system combines a syntactic parser with a semantic model of the domain. MedLEE recognizes negatives which are followed by words or phrases that represent specific semantic classes such as degree of certainty, temporal change or a clinical finding. It also identifies patterns where only the following verb is negated and not a semantic class (i.e. "X is not increased").

The Negfinder system [15] uses a lexical scanner with regular expressions and a parser that uses a restricted context-free grammar to identify pertinent negatives in discharge summaries and surgical notes. It first identifies concepts and then determines whether the concepts are negated. The set of regular expressions is predefined by IT professional based on input obtained from medically trained observers. Their strategy yields over 95% accuracy in classifying negative medical concepts.

NegEx [4] is a simple regular expression-based algorithm that uses a predefined set of pseudo negation phrases, a set of negation phrases, and two simple regular expressions. These phrases are used to filter out sentences containing phrases that falsely appear to be negation phrases, and limits the scope of the negation phrases.

There is no research that tries to learn the negation patterns automatically and then uses the discovered patterns to classify medical concepts that appears in unseen texts.

Physicians are trained to convey the salient features of a case concisely and unambiguously as the cost of miscommunication can be very high. Thus it is assumed that negations in dictated medical narrative are unlikely to cross sentence boundaries, and are also likely to be simple in structure [15]. Based on the above assumptions the purpose of this work is to develop a methodology for learning negative context patterns in medical narratives and measure the effect of context identification on the performance of medical information retrieval.

## 2 Machine Learning Framework

The proposed process begins by performing several pre-processing steps. First all medical documents are parsed. Then all known medical terms are tagged using a tagging procedure presented in our previous work [20]. Finally each text is broken into sentences using a sentence boundary identifier as suggested in [2].

Physician reviewed each document and labeled each medical term indicating whether it appears in positive or negative context. Note that there might be several medical terms in a single sentence not necessarily with the same label. Consider for

instance the compound sentence: "The patient states she had fever, but denies any chest pain or shortness of breath". In this case "chest pain" and "shortness of breath" are negative while "fever" is positive. Thus, this sentence will be represented in the dataset as three different instances – one for each medical term. The resulting labeled dataset is divided into two sets: the training set and the test set.

The training set serves as the input to the learning algorithm. The output of the learning algorithm is a classifier. Given a tagged sentence and a pointer to a tagged term, the classifier classifies the indicated tagged term to either negative or positive context. In this section we present the pattern based learning algorithm used for creating the classifier.

## 2.1 Standard Learning Algorithms

The most straightforward approach is to use existing supervised learning algorithms. In fact the problem presented here is a specific case of a text classification task. The main problem, in comparison to numeric classification tasks, is the additional degree of freedom that results from the need to extract a suitable feature set for the classification task. Typically, each word is considered as a separate feature with either a Boolean value indicating whether the word occurs or does not occur in the document (set-of-words representation) or a numeric value that indicates the frequency (bag-of-words representation).

## 2.2 Regular Expression Learning

In this research we use regular expressions matching as features. Regular expressions hold the expression power of the bag-of-words representation in addition to expressing structure and words distances. From the reasons explained below, instead of using a single regular expression representation for the entire sentence, we are using two regular expressions: one for the string that precedes the targeted medical term and one for the string that follows it. This split may help to resolve some of the problems that arise in compound sentences that include both positive and negative contexts in the same sentence. Recall the example "The patient states she had fever, but denies any chest pain or shortness of breath". In this case the appearance of the verb "denies" after the term "fever" indicates that the term "fever" is left in positive context. The "Bag-of-word" representation of regular expression will be in this case as following: "[^.]{0,200}denies any[^.]{0,200}<DIAGNOSIS>", where the distance 200 is arbitrary determined per the expected sentence length in the domain.

We suggest the following regular expression learning procedure. This procedure consists of three phases.

1. Creating a set of regular expression patterns over the tagged text.
2. Patterns selection – Selecting of the most relevant patterns.
3. Creating a classifier that employs the patterns in a hierarchical manner.

The following subsections discuss each of the above phases.

### 2.2.1 Creation of Patterns

The basis for discovering a regular expression is a method that compares two texts with the same context and incorporates the same concept types (i.e. diagnosis, medication, procedure, etc.). By employing the Longest Common Subsequence algorithm [16] on each part of the sentence (before the targeted term and after the targeted term) a regular expression that fits these two sentences is created. For instance assuming we are given the following two sentences:

The patient was therefore admitted to the hospital and started on Vancomycin as treatments for endocarditis.

The patient was ruled in for myocardial infarction and started Heparin for unstable angina.

The first step is to generalize the sentences by: (a) replacing medical terms with their concept type using the Unified Medical Language System metathesaurus [17]. The UMLS is developed by National Library of Medicine in order to aid the IT systems that help health professionals and researchers retrieve and integrate electronic biomedical information from a variety of sources. For example, “Vancomycin” and “Heparin” are replaced with <MEDICINE>; (b) some of the stop-words can be removed from the text, for example, words such as {the, and, in, ..} might be redundant for the purpose of negation classification patterns, especially in case they are far from the concept; (c) stemming can be applied in order to generalize terms (especially verbs) appearing in the text. For example, the original texts: *denied*, *denying*, *denies* can be replaced with *deny*. Now we can execute the Longest Common Subsequence algorithm on the two normalized sentences as presented in Table 1.

Note that the *Longest Common Substring* algorithm was revised to compare tokens as opposed to comparing characters in its classical implementation. Moreover note that whenever there was only insertion (or only deletion) we added a wild card string of minimum length of 0 and maximum length of the inserted string (including the leading and trailing spaces). On the other hand whenever there was simultaneously insertion and deletion, we added a wild card string with the minimum length of the shortest string and maximum length of the largest string (without leading and trailing spaces because they are part of the common substring).

**Table 1.** Longest Common Substring Generation

| Sentence 1                         | Sentence 2               | Pattern         |
|------------------------------------|--------------------------|-----------------|
| The patient was                    | The patient was          | The patient was |
| therefore admitted to the hospital | ruled in for <DIAGNOSIS> | [^.]{24,35}     |
| and started                        | and started              | and started     |
|                                    | on                       | [^.]{0,4}       |
| <MEDICINE>                         | <MEDICINE>               | <MEDICINE>      |
| as treatments                      |                          | [^.]{0,15}      |
| for <DIAGNOSIS>                    | for <DIAGNOSIS>          | for <DIAGNOSIS> |



As a result of running the Longest Common Subsequence algorithm we can obtain the following pattern. This pattern can now be used to classify concept of type medication appearing in positive contexts.

```
The patient was [^.]{24,35} and
started[^.]{0,4}<MEDICINE>[^.]{0,15}for <DIAGNOSIS>
```

### 2.2.2 Patterns Selection

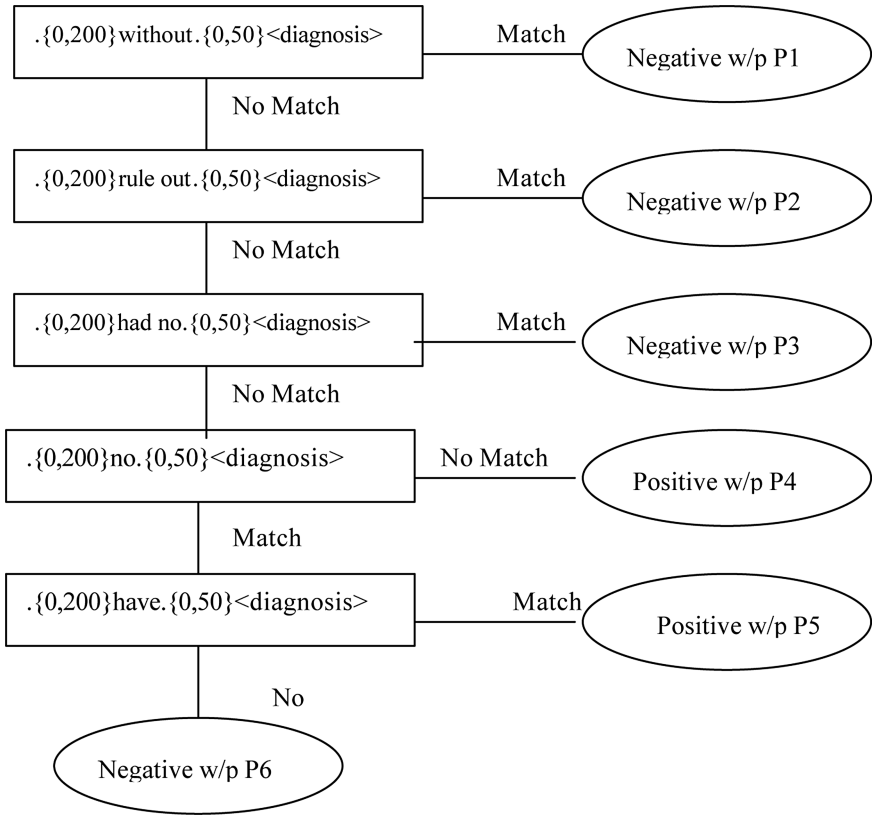
Obviously there are many patterns that can be created (each pair of sentences with the same concept type and context). We tried the following two heuristic procedures sequentially in order to obtain the best discriminating patterns: (a) Eliminating similar patterns— Many of the generated patterns differ only in the distance of important keywords from the seed concept. Grouping such patterns taking only minimum / average / maximum distances eliminates many patterns; (b) Using existing feature selection methods. Feature selection is the process of identifying relevant features in the dataset and discarding everything else as irrelevant and redundant. For this purpose each "regular expression" pattern represents a different feature. In this paper we are using a non-ranker filter feature selection algorithm. Filtering means that the selection is performed independently of any learning algorithm. Non-ranker means that the algorithm does not score each pattern but provides only which pattern is relevant and which is not.

For example, a training set of 140 negative sentences and 140 positive sentences yielded  $140 \times 139 / 2 = 9730$  patterns, 7,145 redundant and trivial patterns were eliminated as described in (a) above. The 2,225 remaining pattern create a dataset of 280 instances with 2225 input binary attributes (0 if pattern does not match sentence; 1 if pattern matches sentence) and target attribute that represent the concept classification ("Positive" or "Negative"). The filter had further reduced the set into 35 'relevant' patterns.

### 2.2.3 Building a Decision Tree Classifier

The filtered matrix, together with the manual classification of each concept, is then fed into a decision tree inducer which creates a classification decision tree.

Using a decision tree as a classifier in this case has several advantages: (1). the sentence is not classified based on a single pattern, but on set of patterns, i.e. this classifier can be used to indicate that a sentence is classified to the label "positive" only if it matched two patterns and does not match to a third pattern. This is more expressive than the classical approach in which the classification is based on a single pattern; (2). The hierarchical structure of decision tree enforces an order (priority) in the usage of patterns, i.e. given a new sentence, not all patterns should be matched in advance but one pattern at a time based on the specific branch traversing. Moreover in this way the desired property of lexical analysis known as unambiguity which is usually resolved by longest match and rule priority is inherently resolved here; (3). As oppose to other classifiers (such as neural networks) the meaning of the classifier can be explained.



**Fig. 1.** Example Decision Tree

An illustrative example of decision tree generated using the C4.5 classifier is presented in Figure 1. It describes a classification decision path where pattern *".{0,200}have.{0,50}<diagnosis>"*, learned from positive examples, indicates a positive context with probability P5 in case the sentence does not match the three (negative) patterns: *".{0,200}without.{0,10}<diagnosis>"*; *".{0,200}rule out.{0,10}<diagnosis>"*; *".{0,200}had no.{0,10}<diagnosis>"* (with probabilities P1, P2, and P3 for negative) but matches the negative pattern *".{0,200}no.{0,50}<diagnosis>"*. Here we denote “negative pattern” as a pattern learned from negative context examples. This demonstrates the power of decision based on matching a sentence with multiple regular expressions.

### 3 Experimental Study

The potential of the proposed methods for use in real word applications was studied. In this experimental study we used a corpus of 565 sentences parsed from de-identified discharge summaries that were obtained from Mount Sinai Hospital in New-York. This same corpus was used in the previous work [20].

A physician was asked to label the each diagnostic term to "positive" or "negative". The sentences corpus was split into two sets: (1) Training set; (2) Test set. We performed several experiments in order to determine the classifier sensitivity to the following parameters: (a) Different training set sizes; (b) the effect of using feature selection (c) the effect of using ensemble of decision trees – it is well known that ensemble can improve accuracy. We examined here in what extent it can improve the results.

### 3.1 Measures Examined

The first measure used is the well-known misclassification rate, indicating the portion of terms that were misclassified by the classifier that was created by the examined algorithm.

Additionally because the identification of the negated is mainly used for improving information retrieval, we will also examine the well-known performance measures precision (P) and recall (R). The notion of "precision" and "recall" are widely used in information retrieval and data mining. Statistics use complementary measures known as "type-I error" and "type-II error".

Precision measures how many cases classified as "positive" context are indeed "positive". Recall measures how many "positive" cases are correctly classified. Usually there is a trade-off between the precision and the recall. Trying to improve one measure often results in a deterioration of the second measure. Thus, it is useful to use their harmonic mean known as F-Measure. In addition we will refer to the accuracy of the classifiers which is the ratio between the correctly classified sentences to the number of sentences in the test corpus.

### 3.2 Algorithm Used

All experiments were performed in the WEKA framework [21]. J48 algorithm is used as the induction algorithm for creating a single decision tree. J48 is a java version of the well-known C4.5 algorithm [19]. For creating the ensemble of decision trees we used the AdaBoost algorithm [8].

In this paper we are using the Correlation-based Feature Subset Selection (CFS) as a subset evaluator [10]. CFS Evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them. Subsets of features that are highly correlated with the class while having low inter-correlation are preferred. The search method used is the Forward Selection Search with Gain Ratio as the scoring criterion.

### 3.3 Results

#### 3.3.1 Overall Results

Table 2 presents the mean F-Measure and Accuracy obtained by the Regular expressions classifier compared to a "Bag-of-words" classifier. The results indicate that the regular expressions classifier outperforms the Bag-of-words method in both F-Measure and Accuracy.

**Table 2.** Benchmark Results

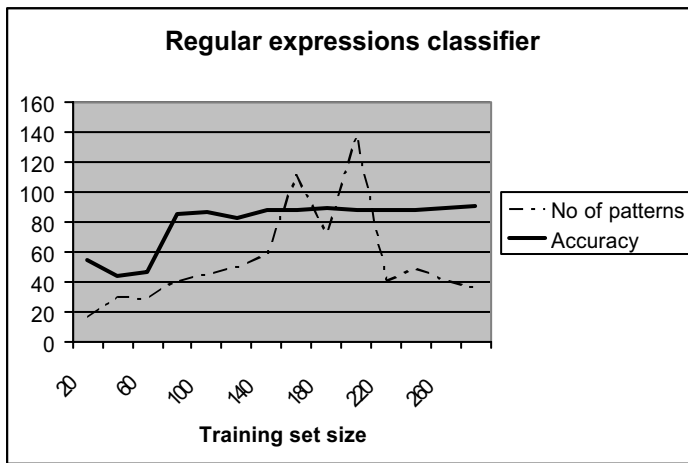
| Method              | Precision | Recall | F-Meas. | Accuracy |
|---------------------|-----------|--------|---------|----------|
| Bag-of-words        | 93.6%     | 85.9%  | 89.75%  | 86.4%    |
| Regular Expressions | 91.6%     | 96.4%  | 93.9%   | 91.77%   |

### 3.3.2 Training Set size

Figure 2 presents a regular expressions classifier training benchmark. It shows how the accuracy converges as the training set size is increased. As shown from the graph, the number of patterns is not directly correlated with the classifier accuracy.

### 3.3.3 The Effect of Feature Selection

Table 3 compares the accuracy obtained by the regular expressions classifier before and after CSF features selection as described in section 3.2 above. The filter eliminates on average 96.5% of the features while achieving substantial accuracy gain. As can be seen in Table 3 results, the filtering improves the classifier accuracy by 5.1% on average. Note that the “bag-of-words” classifier also benefits by applying the CFS feature selection.

**Fig. 2.** Regular expressions Classifier Training**Table 3.** The effect of feature selection

| Classifier mode          |                    | Training set size |       |       |
|--------------------------|--------------------|-------------------|-------|-------|
|                          |                    | 200               | 240   | 280   |
| Before feature selection | Number of features | 2069              | 2300  | 2225  |
|                          | Accuracy           | 81.9%             | 82.8% | 87.0% |
| After feature selection  | Number of features | 138               | 49    | 35    |
|                          | Accuracy           | 88.5%             | 87.7% | 91.8% |

3.3.4 The Effect of Ensemble Size on the Accuracy

Table 4 presents how the ensemble size affects the accuracy. The table presents only the first 5 examined sizes. We also checked the ensemble size up to 10. However it has no effect on the accuracy (it is left as the accuracy of ensemble size of 5). It can be seen that the accuracy of the proposed method is improved from 89.3004% to 91.7695 % (in ensemble size of 3). Nevertheless the accuracy of "Bag-of-words" is only slightly improved by using ensemble of size 4.

3.3.5 The Effect of Combining Discovered Regular Expression with Bag of Words

The results presented in the previous subsections indicated that the proposed method can obtain higher accuracy than the simple "Bag-of-words". In this section we are interesting to check if the improved result implies that the regular expression approach covers all predictive expressiveness of the "Bag-of-words". For this purpose we join the two datasets into a single dataset, thus each instance is now characterized by a set of discovered patterns and by a set of words. Table 5 presents the obtained results. It seems that adding the "Bag-of-words" attributes to the regular expression attributes has reduced accuracy. Moreover this accuracy could be obtained only by increasing the ensemble size to 6 (i.e. it has lower accuracy with higher classifier complexity).

Table 4. The Effect of Ensemble Size on the Accuracy

| Method          | Ensemble Size |           |           |           |           |
|-----------------|---------------|-----------|-----------|-----------|-----------|
|                 | 1             | 2         | 3         | 4         | 5         |
| Reg. Expression | 89.3004%      | 89.3004%  | 91.7695 % | 90.9465 % | 90.9465 % |
| Bag-Of-Words    | 86.0082 %     | 86.0082 % | 85.1852 % | 86.4198 % | 86.4198 % |

Table 5. The Results obtained when Combining Regular Expression with "Bag-of-words"

| Method             | Accuracy  | Ensemble Size |
|--------------------|-----------|---------------|
| Regular Expression | 91.7695 % | 3             |
| Bag-Of-Words       | 86.4198 % | 4             |
| Combined           | 89.3004 % | 6             |

3.3.6 Comparing the Classifier Complexity

As this paper focuses on decision trees, the classifier complexity was measured as the total number of nodes, including the leaves. For multiple decision trees classifiers, the complexity was measured as the total number of nodes in all trees. Table 6 presents the classifier complexity obtained using the regular expression and the bag-of-words as a function of the ensemble size. It can be seen that employing regular expression patterns can reduce the classifier complexity in about 25%.

**Table 6.** The Classifier Complexity as function of the ensemble size

| Method             | Ensemble Size |    |     |     |     |
|--------------------|---------------|----|-----|-----|-----|
|                    | 1             | 2  | 3   | 4   | 5   |
| Regular Expression | 25            | 52 | 81  | 116 | 129 |
| Bag-Of-Words       | 35            | 76 | 103 | 140 | 171 |

### 3.3.7 Analysis of Discovered Patterns

Table 7 presents some of the patterns obtained by the Patterns Learning Algorithm. Previous works showed few words/phrases that are strong indicators of negative context. In these works mostly two word phrases (e.g. “showed no”) were therefore finally considered by the classifier. Terms such as “no” and “not” were not included in their profile because their sole appearance is not a sufficient indication for negation. In this work the pattern learning algorithm learns the two phrase patterns as well as single term patterns such as “no”. This is because the term “no” is a strong negation indicator when it precedes the medical concept, or when combined with additional patterns using a decision tree. These examples explain the accuracy improvement in the pattern learning approach compared to “Bag-of-words” based classification.

**Table 7.** Example of Patterns learned

| Negative Patterns                            |
|--|
| .{0,200} no <DIAGNOSIS>                      |
| .{0,50}showed no.{0,50} < DIAGNOSIS >        |
| Positive Patterns                            |
| .{0,100}patient.{0,5} has.{0,50} <DIAGNOSIS> |
| .{0,100}history of.{0,100} <DIAGNOSIS>       |

## 4 Conclusion

A new pattern based algorithm for identifying context in free-text medical narratives is presented. It has been shown that the new algorithm is superior to “Bag-of-words” classification algorithm. The new algorithm manages to automatically learn patterns similar to manually written patterns for negation detection [15], within the same level of accuracy. The pattern based approach manages to learn and utilize structural features and distances in addition to keywords. Therefore the pattern based approach can be applicable to additional context identification and relationship extraction tasks.

Further work is required to tune and refine the method. Analyzing the reasons for some of the wrong classifications indicates that the method can be improved by adapting the regular expressions to match whole words and ignore partial matches. For example, the pattern “.{0,50}the.{0,50} no.{0,50} <diagnosis>.{0,500}”, mostly negative indicator, matches the positive sentence “*the patient now has <diagnosis> with any movement*” because the word “now” starts with “no”. This improvement can

be easily accommodated. In addition, further study is required to analyze the tradeoffs between the number of patterns and the classifier performance.

Further work is required in order to evaluate the significant of the suggested algorithm in terms of improving IR, especially comparing to the traditional “Bag-of-words” performance for the same IR queries, representing true use-case scenarios. This can be done in the domain of information retrieval from medical narratives as well as additional domains where the negative context is significant for retrieving relevant information.

## References

1. Aronow D, Feng F, Croft WB. Ad Hoc Classification of Radiology Reports. *Journal of the American Medical Informatics Association* 1999; 6(5): 393-411.
2. Averbuch M, Karson T, Ben-Ami B, Maimon O. and Rokach L., Context-Sensitive Medical Information Retrieval, MEDINFO-2004, San Francisco, CA, September 2004, IOS Press, pp. 282-286.
3. Cessie S. and van Houwelingen, J.C. , Ridge Estimators in Logistic Regression. *Applied Statistics* 1997; 41 (1): 191-201.
4. Chapman W.W., Bridewell W., Hanbury P, Cooper GF, Buchanann BG. A Simple Algorithm for Identifying Negated Findings and Diseases in Discharge Summaries. *J. Biomedical Info.* 2001; 34: 301-310.
5. Duda R. and Hart P., *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.
6. Fiszman M., Chapman W.W., Aronsky D., Evans RS, Haug PJ., Automatic detection of acute bacterial pneumonia from chest X-ray reports. *J Am Med Inform Assoc* 2000; 7:593-604.
7. Fiszman M., Haug P.J., Using medical language processing to support real-time evaluation of pneumonia guidelines. *Proc AMIA Symp* 2000; 235-239.
8. Freund Y. and Schapire R. E., Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 325-332, 1996.
9. Friedman C., Alderson P, Austin J, Cimino J, Johnson S. A General Natural-Language Text Processor for Clinical Radiology. *Journal of the American Medical Informatics Association* 1994; 1(2): 161-74.
10. Hall, M. Correlation- based Feature Selection for Machine Learning. Ph.D. Thesis, University of Waikato, 1999.
11. Hersh WR, Hickam DH. Information retrieval in medicine: the SAPHIRE experience. *J. of the Am Society of Information Science* 1995; 46:743-7.
12. Hripcsak G, Knirsch CA, Jain NL, Stazesky RC, Pablos-mendez A, Fulmer T. A health information network for managing innercity tuberculosis: bridging clinical care, public health, and home care. *Comput Biomed Res* 1999; 32:67-76.
13. Keerthi S.S., Shevade S.K., Bhattacharyya C., Murth K.R.K., Improvements to Platt's SMO Algorithm for SVM Classifier Design. *Neural Computation* 2001; 13(3):637-649.
14. Lindbergh D.A.B., Humphreys B.L., The Unified Medical Language System. In: van Bommel JH and McCray AT, eds. 1993 Yearbook of Medical Informatics. IMIA, the Netherlands, 1993; pp. 41-51.
15. Mutalik P.G., Deshpande A., Nadkarni PM. Use of general-purpose negation detection to augment concept indexing of medical documents: a quantitative study using the UMLS. *J Am Med Inform Assoc* 2001; 8(6): 598-609.

16. Myers E., An  $O(ND)$  difference algorithm and its variations, *Algorithmica* Vol. 1 No. 2, 1986, p 251.
17. Nadkarni P., Information retrieval in medicine: overview and applications. *J. Postgraduate Med.* 2000: 46 (2).
18. Pratt A.W. Medicine, computers, and linguistics. *Advanced Biomedical Engineering* 1973: 3:97-140.
19. Quinlan, J. R. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
20. Rokach L., Averbuch M., Maimon O., *Information Retrieval System for Medical Narrative Reports*, *Lecture Notes in Artificial intelligence* 3055, pp. 217-228 Springer-Verlag, 2004.
21. Witten I. H. and Frank E. *"Data Mining: Practical machine learning tools and techniques"*, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.



# A Hybrid Method for Speeding SVM Training

Zhi-Qiang Zeng<sup>1</sup>, Ji Gao<sup>2</sup>, and Hang Guo<sup>3</sup>

Department of Computer Science and Engineering, Zhejiang University,  
310027 Hangzhou, China

<sup>1</sup>lbxzzq@hotmail.com

<sup>2</sup>gaoji@mail.hz.zj.cn

<sup>3</sup>hangguo@zju.edu.cn

**Abstract.** Support vector machine (SVM) is a well-known method used for pattern recognition and machine learning. However, training a SVM is very costly in terms of time and memory consumption when the data set is large. In contrast, the SVM decision function is fully determined by a small subset of the training data, called support vectors. Therefore, removing any training samples that are not relevant to support vectors might have no effect on building the proper decision function. In this paper, an effective hybrid method is proposed to remove from the training set the data that is irrelevant to the final decision function, and thus the number of vectors for SVM training becomes small and the training time can be decreased greatly. Experimental results show that a significant amount of training data can be discarded by our methods without compromising the generalization capability of SVM.

## 1 Introduction

Support vector machine (SVM) is one of the most powerful classification techniques that was successfully applied to many real world problems [1,2]. However, training a SVM involves solving a constrained quadratic programming (QP) problem, which requires large memory and takes enormous amounts of training time for large-scale applications [3]. On the other hand, the decision function constructed by SVM is dependent only on a small subset of the training data called support vectors that lie close to the decision boundary. Therefore, if one knows in advance which patterns correspond to the support vectors, the same solution can be obtained by solving a much smaller QP problem that involves only the support vectors. The problem is then how to select training examples that are likely to be support vectors. Recently, there has been considerable research on data selection for SVM training. Techniques include: Random Selection [4,5], Clustering analysis [6,7], Bagging [8], and Rocchio Bundling [9]. Random Sampling and Rocchio Bundling could over-simplify the training set, hence losing the benefits of the using SVM. In bagging technique, the testing process becomes very expensive [9]. As for clustering analysis, the algorithm proposed in [6] and [7] works for the linear kernel only and uses the fact that the original space and the feature space are the same under a linear kernel.

In this paper, we present an effectively hybrid method to speed up the training process by reducing the number of training data, which is applicable to both the linear

kernel and the nonlinear kernel. This method first applies the  $k$ -mean algorithm to organize positive and negative training data respectively in clusters. After identifying the initial clusters, the method starts training a reformulated SVM only from the representatives of these clusters and generates an approximated decision boundary. Then, clusters away from the rough boundary are eliminated and clusters near to this plane are refined to further remove non-relevant samples in deciding resulting SVM classifier. Consequently, the number of training vectors is decreased greatly, speeding up the SVM training and requiring less computational efforts without degrading the result.

This paper is organized as follows. Section 2 gives a brief introduction to the theoretical background with reference to classification principles of SVM. Section 3 proposes the new method that selects the potential support vectors for training SVM. Experimental results are demonstrated in Section 4 to illustrate the efficiency and effectiveness of our algorithm. Conclusions are included in Section 5.

## 2 Support Vector Machines

Support vector machines (SVMs) [1,2] have been a useful classification method. Given training vectors  $x_i \in R^n$ ,  $i=1, \dots, l$  in two classes, and a vector  $y \in R^l$  such that  $y_i \in \{1, -1\}$ , the support vector technique requires the solution of the following optimization problem:

$$\begin{aligned} \min_{w, b, \xi} \quad & \frac{1}{2} w^T w + C \sum_{i=1}^l \xi_i, \\ \text{subject to} \quad & y_i (w^T \phi(x_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, i = 1, \dots, l. \end{aligned} \quad (1)$$

where  $\xi_i$  for  $i=1, \dots, l$  are slack variables introduced to handle the non-separable case. The constant  $C > 0$  is the penalty parameter that controls the trade-off between the separation margin and the number of training errors. Using the Lagrange multiplier method, one can easily obtain the following Wolfe dual form of the primal quadratic programming problem:

$$\begin{aligned} \min_{\alpha_i, i=1, \dots, l} \quad & \frac{1}{2} \sum_{i, j=1}^l \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle - \sum_{i=1}^l \alpha_i, \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C, i = 1, \dots, l, \\ & y^T \alpha = 0. \end{aligned} \quad (2)$$

Solving the dual problem, one obtains the multipliers  $\alpha_i$ ,  $i=1, \dots, l$ , which give  $w$  as an expansion

$$w = \sum_{i=1}^l \alpha_i y_i x_i. \quad (3)$$

According to the Karush-Kuhn-Tucker (KKT) optimality conditions, we have

$$\alpha_i = 0 \Rightarrow y_i (\langle w, x_i \rangle + b) \geq 1 \text{ and } \xi_i = 0,$$

$$\begin{aligned}
0 < \alpha_i < C &\Rightarrow y_i(< w, x_i > + b) = 1 \text{ and } \xi_i = 0, \\
\alpha_i = C &\Rightarrow y_i(< w, x_i > + b) \leq 1 \text{ and } \xi_i \geq 0.
\end{aligned} \tag{4}$$

Therefore, only a number of Lagrange multipliers will be nonzero, and the data points associated with them refer to support vectors. A key property of support vector machine is, that training a SVM on the support vectors alone gives the same result as training on the complete example set. All the remaining  $\alpha_i$  are zero and the corresponding training examples are irrelevant to the final solution.

In practice, linear decision functions are generally not rich enough for pattern separation. The basic idea is to map the input data  $x$  into a higher dimensional feature space  $F$  via a nonlinear mapping  $\phi$  and then a linear classification problem is obtained and solved in this feature space. This is achieved implicitly by using different type of symmetric functions  $k(x_i, x_j)$  instead of the inner products  $< x_i, x_j >$ . Consequently, the optimal decision boundary in the feature space  $F$  represents a nonlinear decision function of the form

$$f(x) = \text{sgn}\left(\sum_{i=1}^l \alpha_i y_i k(x_i, x) + b\right). \tag{5}$$

### 3 Training Data Selection for Support Vector Machine

In this section, the hybrid method to accelerate training speed would be introduced in detail.

#### 3.1 Train a Reformulated SVM

At first,  $k$ -mean clustering method is employed to generate a certain number of clusters from positive and negative datasets independently. Given  $N$  data points in a cluster:  $\{x_i\}$  where  $i=1, 2, \dots, N$ , the center  $O$  and radius  $R$  of the cluster are defined as:

$$O = \frac{\sum_{i=1}^N x_i}{N}, R = \left(\frac{\sum_{i=1}^N \|x_i - O\|^2}{N}\right)^{\frac{1}{2}}. \tag{6}$$

The cluster centers would well reflect the relative position of point-clusters of input dataset and thus characterize the outline of the full dataset. Therefore, an approximated decision boundary can be constructed by training a SVM from the cluster centers. It is noted that each center represents a different number of samples within the corresponding cluster, currently. Thus, we reformulate the objective function of the standard soft-margin SVM by assigning an adaptive penalty term to outliers according to the number of samples represented by the center. As a result, the objective function of SVM is reformulated as following:

$$\begin{aligned}
\min_{w, b, \xi} \quad & \frac{1}{2} w^T w + C \sum_{i=1}^n \lambda_i \xi_i, \\
\text{subject to} \quad & y_i (w^T \phi(x_i) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, i = 1, \dots, n.
\end{aligned} \tag{7}$$

According to KKT condition, we can express (7) as a dual problem given next,

$$\begin{aligned} \min_{\alpha_i, i=1, \dots, k} \quad & \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j k < x_i, x_j > - \sum_{i=1}^n \alpha_i, \\ \text{subject to} \quad & 0 \leq \alpha_i \leq \lambda_i C, i = 1, \dots, n, \\ & y^T \alpha = 0. \end{aligned} \quad (8)$$

where  $n$  is the number of clusters and  $\lambda_i$  for  $i=1, \dots, n$  are weights introduced to adaptively penalize the outliers. Here,  $\lambda_i$  is defined as (Corresponding with the positive and negative training data respectively).

$$\lambda_i = \frac{\# \text{ of points within } C_i}{\# \text{ of positive(negative) training data} / n_+ (n_-)}, \quad i = 1, \dots, n, \quad (9)$$

$$n_+ + n_- = n. \quad (10)$$

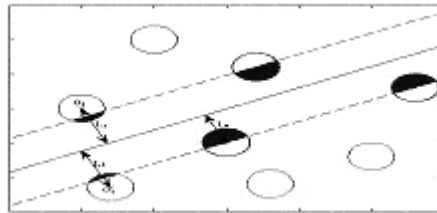
where  $n_+$  and  $n_-$  are the number of clusters corresponding with the positive and negative training data respectively, and  $C_i$  denotes the  $i^{\text{th}}$  cluster. In section 4, experimental results show that with applying the reformulated SVM instead of the standard SVM in the stage, the resulting SVM classifier has a higher precision.

### 3.2 Identify the Low Margin Clusters

With the approximated decision boundary  $h$ , we determine the *low margin clusters* [6] that are close to the boundary  $h$ . Let *support clusters* be the clusters whose center points are the support vectors of the boundary  $h$ . Let  $L_s$  be the distance from the boundary  $h$  to the center of a support cluster, and let  $L_i$  be the distance from the boundary  $h$  to the center of a cluster  $E_i$ . Then, we consider a cluster  $E_i$  which satisfies (11) as a *low margin cluster* (illustrated in Figure 1, marked with black color).

$$L_i - R_i < L_{ms}. \quad (11)$$

where  $R_i$  is the radius of the cluster  $E_i$  and  $L_{ms}$  is the maximum distance of all  $L_s$ .



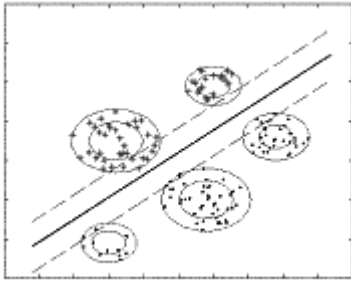
**Fig. 1.** Examples of the low margin clusters

Because the soft-margin SVM generates the support vectors having different distances to the boundary, we employ  $L_{ms}$  instead of  $L_s$  in (11).

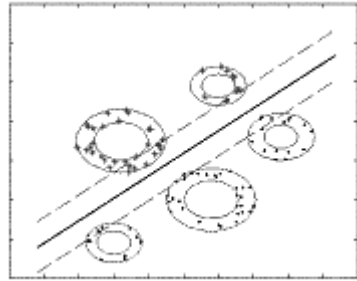
According to SVM theory, samples close to the decision boundary have a higher likelihood of being a support vector and the samples far from decision boundary have less effect when identifying support vectors. Based on this rule, the samples within the *low margin clusters* are more likely to be support vectors because they are closer to the approximated decision boundary  $h$  than other samples. As a result, *low margin clusters* only are preserved and those clusters that do not satisfy (11) are discarded in order to reduce the size of huge training sets without degrading the results.

### 3.3 Identify the Circle Region

After removing those clusters that are far from the approximated decision boundary  $h$ , only the *low margin clusters* remain to represent the whole training set.



**Fig. 2.** Examples of inner and outer clusters



**Fig. 3.** Circle region

From the Figure 2, it can be seen that samples close to the *low margin cluster* boundary have a high probability of affecting the support vector selection. Therefore, it is better to keep a *circle region* close to the cluster boundary, so that no samples would be removed from that region. This region is defined based on the cluster radius or number of samples within the cluster. The outcome would be an inner cluster, where samples within are far from the decision boundary. Therefore, it might have no effect on building the proper decision function to remove those samples. Samples between the inner cluster and the outer cluster are the samples within the *circle region* illustrated in Figure 3 and are not to be removed. To determine samples within the *circle region*, for each sample  $x_i$  within the *low margin cluster*  $E$ , we compute the distance  $r_i$  from the  $x_i$  to the cluster center and sort the samples according to the corresponding value of  $r_i$ , and then remove a percentage samples with the smallest value of  $r_i$ . The percentage threshold is defined as:

$$\text{Percentage threshold} = \frac{\text{samples within the inner cluster}}{\text{all samples within the low margin cluster}} \times 100\% . \quad (12)$$

### 3.4 The Main Algorithm

With the above preprocessing steps, only these samples within the *circle region* are picked out to generate the reduced sample set whose size is much smaller than that of the whole training set. The reduced set is then used to construct the resulting SVM classifier. The sketch of our method can be summarized as follows:

*Input:* positive dataset  $D^+$ , negative dataset  $D^-$

*Output:* a boundary function  $b$

*Function:*

Cluster( $D$ ): Return a set of clusters from a dataset  $D$

getClusterCenter( $C$ ): return a set of centers from a set of clusters  $C$

getLowerMargin( $b'$ ,  $C$ ): return the *low margin clusters* from a set of clusters  $C$  which are close to the boundary  $b'$

getCircleRegion( $C'$ ): return a set of *circle regions* from a set of clusters  $C'$

*Algorithm:*

1.  $C_p := \text{Cluster}(D^+)$ ;  $C_n := \text{Cluster}(D^-)$ ;
2.  $D_{\text{initial}} := \{ \text{getClusterCenter}(C_p) \cup \text{getClusterCenter}(C_n) \}$ ;
3.  $b' := \text{SVM.train}(D_{\text{initial}})$ ; // construct the initial boundary  $b'$
4.  $C' := \text{getLowerMargin}(b', \{C_p \cup C_n\})$ ;
5.  $D_{\text{reduced}} := \text{getCircleRegion}(C')$ ;
6.  $b := \text{SVM.train}(D_{\text{reduced}})$ ;
7. Return  $b$ ;

## 4 Experiments and Discussions

To verify the effectiveness and efficiency of the proposed algorithm, we classified two large datasets by our method and the method that training a full QP problem, respectively, and then compared the test results.

### 4.1 Experiment Setup

The LIBSVM [10] is used for standard SVM implementation, and the reformulated SVM is implemented based on the modification of the LIBSVM. For two datasets, Gaussian kernels which are nonlinear kernels with default variance setting in LIBSVM were used and the parameter  $C$  in (1) is 100. All our experiments are done in a computer with P4 1.8 GHZ and 1 Gigabytes RAM. The operating system is Windows 2003 Server.

The two large datasets “adult (14 features)”, “forest (54 features)” are selected from the UCI Machine Learning repository [11] to test our method. For the “forest” dataset, cove type 5 is considered for the classification and 70,000 samples were selected and randomly divided into two groups of 50,000 and 20,000, respectively. One such set is considered as the training set and the other as the test set. For the “adult” dataset, we use the standard training and testing sets, which have 32561 samples and 16281 samples, respectively. The parameter values of  $k$ -mean on “adult”

are 50 and 20 that correspond to positive and negative training data respectively. For the “forest” dataset, the corresponding values are 40 and 65.

4.2 Performance of Our Method

Results obtained with two datasets (Table 1), show that it is possible to reduce the training set even up to 90%, without significant effect upon the classification results. Since only a few samples were remaining, the time taken for the calculations is minimal. Even considering the preprocessing cost, the total spending time is much less than that for training SVM with the complete training set. Besides, under some percentage thresholds, the classification rate is even higher than that for training a full QP problem, indicating the increased generalization of our method. On the other hand, less important support vectors were eliminated in building the margins. Employing fewer support vectors results in a fast evaluation of the discriminant hyperplane in the future use of SVM. All this happened while the generalization ability was not degraded. Compared to training a SVM from the entire dataset, our method is able to generate more efficient SVM with similar classification rate.

Table 1. The comparison of computation and precision under different percentage thresholds

| Data set |             | Percentage threshold (%) | # of training vectors | # of support vectors | Testing accuracy (%) | Cpu time (preprocess + SVM)(s) |
|----------|-------------|--------------------------|-----------------------|----------------------|----------------------|--------------------------------|
| Adult    | Reduced set | 75                       | 5730                  | 2917                 | 84.98                | 144                            |
|          |             | 80                       | 4592                  | 2336                 | 84.93                | 138                            |
|          |             | 85                       | 3443                  | 1821                 | 83.85                | 133                            |
|          |             | 90                       | 2290                  | 1203                 | 83.27                | 130                            |
|          | Full set    |                          | 32561                 | 12273                | 84.54                | 1661                           |
| Forest   | Reduced set | 75                       | 12482                 | 6465                 | 87.56                | 1061                           |
|          |             | 80                       | 10002                 | 5297                 | 87.28                | 945                            |
|          |             | 85                       | 7499                  | 4038                 | 87.27                | 875                            |
|          |             | 90                       | 4996                  | 2684                 | 87.12                | 841                            |
|          | Full set    |                          | 50000                 | 23915                | 87.36                | 8818                           |

4.3 Influence of the Percentage Threshold

From Table 1, it can be concluded that the percentage threshold provides a trade-off between the training speed and the classifying accuracy. When it is big, a large number of data points will be removed so that the training has a high speed but with a large classifying error. In reverse, a small threshold means a good classification performance but with a slow training. However, the gap of classifying accuracy between the best and the worst is very slight when threshold value is bigger than 75%. In practical applications, we can select a big threshold to deeply reduce the training time without great influence on classification accuracy.

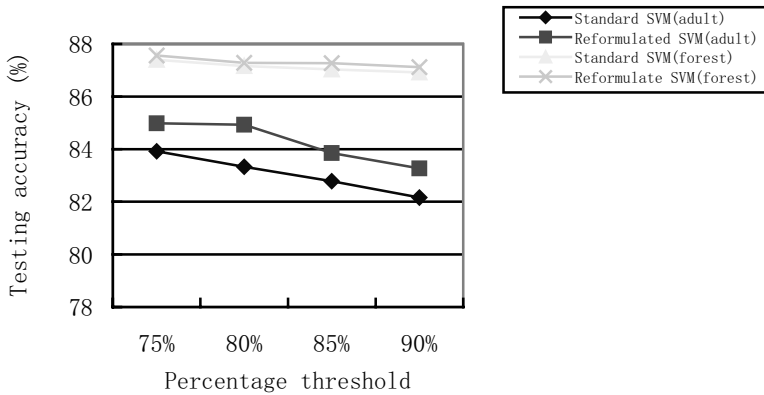
#### 4.4 Effect of the Reformulated SVM

Define  $h$  and  $h'$  to be approximated boundaries that are generated by training the reformulated SVM and the standard SVM from the cluster centers in step 2 of our method, respectively. The classifying rate of the boundaries  $h$  and  $h'$  on test sets is reported in Table 2.

**Table 2.** The comparison of precision of the approximated boundaries constructed by training different SVM

| Data set | Testing accuracy<br>( standard SVM ) (%) | Testing accuracy ( reformulated<br>SVM ) (%) |
|----------|--|--|
| Adult    | 76.3                                     | 80.50  |
| Forest   | 78.4                                     | 81.73  |

It can be seen that the precision of  $h$  is high compared with that of  $h'$ , which proves that  $h$  lies much closer to the desired boundary than  $h'$ . Therefore, the resulting SVM classifiers will have a higher classifying accuracy if we employ  $h$  instead of  $h'$  in step 2. Experimental results on test sets illustrated in Figure 4 also verify the opinion.



**Fig. 4.** Precision of resulting SVM classifiers with training a reformulated SVM in step 2 VS that with training a standard SVM in the same stage

## 5 Conclusions

A hybrid method to overcome the problems with huge training set for SVM, is investigated in this paper. In the proposed technique, the behavior of selecting support vectors was taken into account in attempting to minimize the effects on the final result caused by reducing the training set. In this way, we induce the support vectors, the description of the boundary, as fine as possible while keeping the total number of training data points as small as possible. Through two simulations, we obtained



promising results: SVM with the selected patterns was trained much faster with fewer redundant support vectors, without compromising the generalization capability. Experiment results also show that the proposed method is able to solve problems with the nonlinear kernel.

## References

1. C. Cortes and V. Vapnik. "Support-vector network". *Machine Learning*, 20:273–297, 1995.
2. V. Vapnik. "Statistical Learning Theory. Wiley", New York, NY, 1998.
3. T. Joachims. "Making large-scale SVM learning practical". In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, Cambridge, MA, 1998. MIT Press.
4. J. L. Balcazar, Y. Dai and O. Watanabe, "Provably Fast Training Algorithms for Support Vector Machines", in *Proc. of the 1<sup>st</sup> IEEE International Conference on Data mining*, IEEE Computer Society (2001) pp.43-50.
5. D. K. Agarwal, "Shrinkage estimator generalizations of proximal support vector machines", in *Proc. of the 8<sup>th</sup> ACM SIGKDD international conference of knowledge Discovery and data mining*, Edmonton, Canada, 2002.
6. Yu, H., Yang, J., Han, J.: "Classifying large data sets using svms with hierarchical clusters". In: *Proc. ACM SIGKDD*. (2003) 306-315.
7. B. Daniael and D. Cao, "Training Support Vector Machines Using Adaptive Clustering", in *Proc. Of SIAM International Conference on Data Mining 2004*, Lake Buena Vista, FL, USA.
8. G. Valentini and T.G. Dietterich, "Low Bias Bagged Support Vector Machines", in *Proc. of the 20<sup>th</sup> International Conference on Machine Learning ICML 2003*, Washington D.C. USA, pp. 752-759.
9. L. Shih, Y D.M Rennie, Y. Chang, and D.R. Karger, "Text Bundling : Statistics-based Data Reduction", in *Proc. of the Twentieth International Conference on Machine Learning (ICML-2003)*, Washington DC, 2003.
10. C.-C. Chang and C.-J. Lin.: "LIBSVM: a library for support vector machines", 2001. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
11. Murphy, P. M., & Aha, D.W. (1994). UCI repository of machine learning databases. Irvine, CA (Available at <http://www.ics.uci.edu/~mllearn/MLRepository.html>)

# Design of Service Management System in OSGi Based Ubiquitous Computing Environment

Seungkeun Lee and Jeonghyun Lee

Department of Computer Science and Engineering  
Inha University, Incheon, Korea  
sglee@nlsun.inha.ac.kr, jhlee@inha.ac.kr

**Abstract.** The OSGi offers a unique opportunity for ubiquitous computing as a potential framework for achieving interoperability between various sensors, home appliances, and networked devices. The OSGi framework supports a remote installation of a bundle, which is a unit that installs and deploys services and service matching method based on service syntax. However, in order for the service in ubiquitous computing environment, a specific form of bundle such a mobile service manager is needed, one which is able to move through a heterogeneous network. And more precise service matching mechanism is needed, one which is able to search services using semantic description. This paper proposes a method that can manage bundles for supporting dynamic service's mobility between frameworks, in order to ensure the mobility of services in a multiple the OSGi framework environment and a method that can search service using extended OWL-S. This service management system we are proposing implements a bundle form which can perform in an OSGi framework as well as manage the mobile services and semantic service matching mechanism. As a result, service management in a ubiquitous computing environment will be supported more efficiently.

**Keywords:** OSGi, Ubiquitous Computing, Service Management.

## 1 Introduction

OSGi is an industry plan regarding the standards for allowing sensors, embedded computing devices and electronic appliances to access the Internet, and it provides a Java-based open standard programming interface for enabling communication and control between service providers and devices within home and small business networks[1,2]. Whereas previous technologies focused on interoperation among devices, OSGi places emphases on service delivery, allocation and management for the devices. Furthermore, linkage services concerning Jini or UPnP can be deployed or interacted based on development of OSGi-based applications. OSGi is already providing connections for various devices in local networks such as the home, office and vehicle and providing manageable and expandable frameworks, expediting the arrival of the ubiquitous computing environment. Moreover, by defining standard execution environments and service interfaces, OSGi enhances dynamic findings and collaboration among various heterogeneous resources[3][4].

An OSGi-based ubiquitous computing system has a structure for distributing new services, and the structure consists of only the elements on the local network, giving it relatively closed characteristics. However, while service management and distribution can be dynamically executed within such single OSGi framework, there is insufficient support for applications with mobility among multiple frameworks. And according to moving service among multiple frameworks, more precise service matching mechanism is needed. This enables ubiquitous services to be accessed by contents rather than by keywords. Services can be discovered, selected and composed automatically by other services.

Therefore, there must be sufficient consideration for mobility of the users, devices and sensors among multiple OSGi frameworks in the expanded smart space, calling for research efforts in services supporting such mobility and service matching using service semantics. Since the user, devices and sensors have mobility in a smart space, the services need to be mobile with their execution statuses stored. Also, mobility management must be efficiently supported for such service elements in the expanded smart space where multi-dimensional OSGi frameworks are located.

For example, let's say a user is heading toward home while listening to an mp3 music files on the PDA. Upon arrival at home, if the user wishes to listen to the same music on the PC, there is a cumbersome task of having to select the music play list from the PC directory. Even if there is an identical music play list in the PC, which eliminates the need for the user to select each individual song, there would not be the satisfaction of continuously listening to the music that had been playing on the PDA. However, if the mp3 player can be moved from the PDA to the PC, the music play list and song information can be maintained, allowing the user to appreciate the music without interruption[5][6][7].

This study deals with designing an OSGi-based framework using a service mobility technology that supports mobility and duplication with status information in the distribution environment and semantic service matching and composition. By supporting bundles in the form of a mobile agent, the designed framework also supports mobility of the bundles within multiple OSGi system environments. Therefore, it can support mobility of various elements such as services for specific components or users as well as device drivers. In order to do so, the OSGi framework's open source Knopflerfish 1.3.3 is expanded and a mobile agent management system is designed to support the bundles' mobile lifecycle[8][9].

## 2 Relative Works

In this chapter, we describes the OSGi framework and OWL-S which is standard for service semantic description.

### 2.1 OSGi (Open Services Gateway Initiative)

OSGi is a non-profit organization that defines standard specifications for delivering, allocating and managing services in the network environment. In its initial stage, OSGi was focused on the home service gateway, but it has recently expanded from a specific network environment to the ubiquitous environment. In turn, the objective of

OSGi has become implementation of the service gateway for diverse embedded devices and their users[10].

The OSGi framework signifies the execution environment for such services and includes the minimum component model, management services for the components and the service registry. A service is an object registered in a framework used by other applications. For some services, functionality is defined by the interfaces they implement, allowing different applications to implement identical “service” types. The OSGi framework installs an OSGi component called a “bundle” and supports a programming model for service registration and execution. The OSGi framework provides the mechanism for managing the bundle lifecycle. If a bundle is installed and executed in a framework, it can provide its services. Moreover, it can find and use other services available on the framework and can be grouped with other bundle services through the framework’s service registry. When registering a service in the OSGi framework’s service registry, a bundle stores the service attributes in the form of the attribute-value pair. Such service attributes are used so that they can be distinguished among multiple service providers.

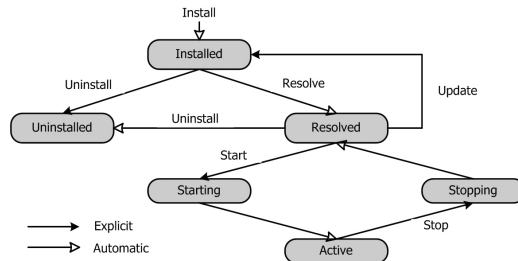


Fig. 1. The lifecycle of the bundle

## 2.2 OWL-S

OWL-S is an OWL ontology for describing services[11]. The Language OWL represents part of the Semantic Web initiative to provide semantics on the Internet, and to make web content less ambiguous. Therefore OWL-S enable a service to discover, request, select, and compose other services automatically.

OWL-S is published by the W3C, which is based on DAML-S using DAML-OIL ontology developed by DARPA. And, OWL-S is designed as an upper layer to SOAP, WSDL, WSFL, XLANG, BP4WS(Business Process Execution Language for Web Service). As an OWL ontology, OWL-S retains all the benefits of web contents described by OWL. Namely, it has a well-defined semantics, it enables the definition of a service vocabulary in terms of objects and complex relationships between them, including classes, subclass relationships, and cardinality restrictions. It also includes all the XML type information. OWL-S is composed by ServiceProfile, ServiceModel and ServiceGrounding. These properties represent the three features which are presents, describedBy, support. ServiceProfile is similar to a yellow page for a service. It describes properties of a service necessary for automatic discovery, such as the functionality the servi+ce offers, and its inputs, outputs, preconditions and effects.

ServiceModel describes the service's process model. It is designed to enable automated composition and execution of services. ServiceGrounding connects the process model description to communication-level protocols and message descriptions in WSDL. In OWL-S, the QoS ontology used to estimate the service is not defined. The QoS ontology includes the properties regarding response time, execution cost, and the reliability of service.

### 3 Design of Service Management System

Service management system consists of service mobility manage and service discovery manager. Figure 2 describes service management system. The overall structure consists of service mobility manager for managing mobility, service discovery manager for service matching and composition based on service semantic and ontology inference engine.

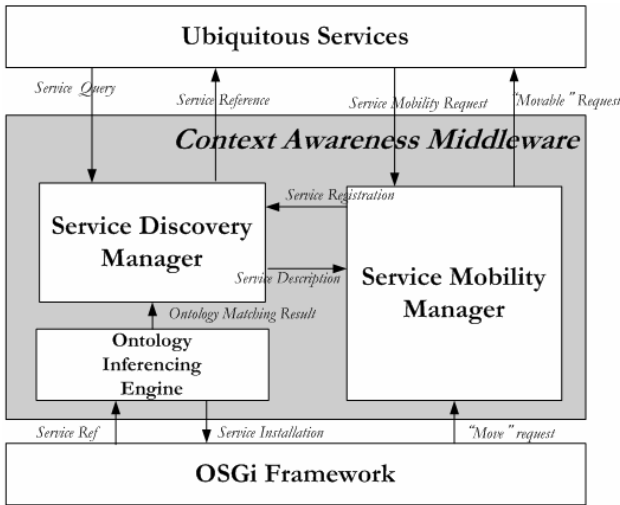


Fig. 2. The structure of the Service Mobility Manager

If service is moved new OSGi framework, service description is registered in service discovery manager. This service description is used in service matching and generation of service composition plan.

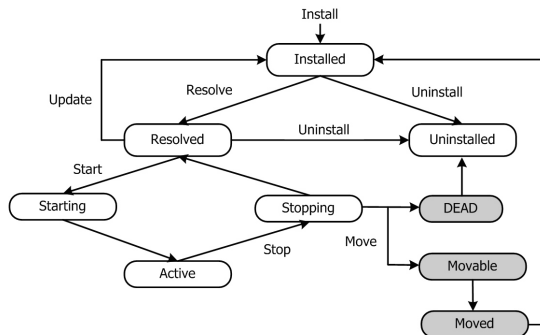
#### 3.1 Service Mobility Manager

Service Mobility Manager consists of Mobility Interface, Service Serializer, SOAP Manager. Fig. 3. describes Service Mobility Manager. The overall structure consists of the mobile interface for managing mobility, elements for processing serialization/deserialization and elements for SOAP message transmission/reception

When the Mobility Interface receives a mobility request from a bundle service, it manages the service bundle's lifecycle. The status information prior to mobility is marshalled into XML by the ServiceSerializer, and the SOAPClient delivers the SOAP message to the destination. The class file is installed through the destination BundleInstaller, and the ServiceDeserializer resumes the service by unmarshalling the SOAP message into an object.

### The Extended Lifecycle for Service Bundle

In this paper, we extend the status of bundle in OSGi which compose of 'Resolved', 'Starting', 'Active' and 'Stopping', adding 'DEAD', 'Movable', 'Moved'. 'Dead' status is different from 'Uninstalled' which means that bundles are omitted automatically. 'DEAD' is used for checking information of service before it is received move request. The status of bundle is changed 'Movable' to 'Moved' for the service which is received move request. 'Move' status is used for the process which OSGi framework sends service to other OSGi framework. If OSGi framework would finish sending of service to other OSGi framework, service status is changed to 'Uninstalled', and this bundle is removed.



**Fig. 3.** The extended lifecycle for service bundle

### Mobility of Service

Upon receipt of a mobility request, the service is switched to the mobility request state and execution suspension is requested. The service receiving the request returns after completing the action currently in process. If converted to the mobile state, the status information is serialized into the XML format using the Serializer, and service mobility is requested to the SOAPClient. The SOAPClient verifies the destination and generates an SOAP message to call SOAPService to the destination. If mobility is successful, the service currently being executed is deleted from the registry.

At the destination, the SOAPService waiting for the SOAP message receives the URL information regarding the class location as well as the serialized data and delivers them to the MobileBundleManager. Prior to deserializing the received object, the MobileBundleManager installs the bundle from a remote location through the

BundleInstaller. Upon successful installation, the object is deserialized and restored to the state prior to mobility. Finally, the service is converted to the RUNNABLE state and registered at the service registry. Algorithm 1 describes the process of service transmission between OSGi frameworks.

**Algorithm 1.** Sending ServiceObject with ServiceID

```

Input
ServiceID : ServiceID registered in Service
RegistryVariables
ServiceRef : Service Reference
ServiceDes : Service Description
ServiceStatus : Service Status Information
SOAPMessage : SOAPMessage used in Service Transmission
Begin Algorithm
    ServiceRef =
    ServiceManager.ServiceFinder.GetServiceRef(ServiceID)
    ServiceDes =
    ServiceManager.ServiceFinder.GetServiceDes(ServiceID)
    res = ServiceRef.beforeMoving()
    If (res is true)
    Begin If
        ServiceStatus = ServiceSerializer.serializer(res)
        SOAPMessage = SOAPService.makeSOAPMessage(URL,
            ServiceStatus, ServiceDes)
        sendMessage(TargetURL, SOAPMessage)
    End If
End Algorithm

```

When moving an object, it was sent to the SOAP Body element was a parameter by serializing it into an XML format as below. Castor and Apache Axis were used for serialization and SOAP transmission, respectively. Other resource and class files were not sent as attached files to the SOAP message. Rather, a mobile agent bundle was installed using the bundle allocation function at the remote location.

The example of the transferring SOAP message

```

<soapenv:Envelope ....>
  <soapenv:Body>
    <ns1:service xmlns:ns1="http://webserivce.ema">
      <obj xsi:type="xsd:string">
        <lt;agent>
          <lt;status>5</status>
          <lt;action>
            <lt;name>Simple Action</name>
            <lt;msg>goodluck</msg>
          </action>
        </agent></obj>
      </ns1:service>
    </soapenv:Body>
  </soapenv:Envelope>

```

The procedure and configuration for creating mobile bundles using the framework implemented in this paper are as follows.

First, in order to implement a user-defined agent class, an XML schema must be created afterwards for object serialization using Castor in a format where the agent class defined in the agent bundle is inherited.

The Manifest file of the user-defined agent designated the Bundle-Activator as the MobileBundleActivator, for which the ema.core.activator package in the agent bundle was imported. The MobileBundleActivator class reads the Agent-Class and Agent-Name header values indicated in the Manifest using the bundle objective and registers them in EAR. In order to register a user agent as a service from the AgentManager bundle, the agent class registered in EAR must be dynamically loaded. For this purpose, the user-defined agent was limited to the agent.impl package.

The example of the Manifest file

```
Bundle-Activator:ema.core.activator.
    MobileBundleActivator
Agent-Name: MyAgent
Agent-Class: agent.impl.MyAgent
Import-Package: org.osgi.framework,
    ema.core.activator
...
```

### 3.2 Service Discovery Manager

Service discovery manager presents service matching service which enable service to search other services using semantic description and service composition which can generate the execution plan of service for service request.

#### Service Ontology for Service Matching and Composition

In section 2, we described an overview of OWL-S. In this paper, we designed a ontology model, the extends of OWL-S for this framework. Figure 3 describes this ontology model. The QoS ontology is composed of three properties. These are a response time, a execution cost, and the reliability of service. Each property has three values, min/average/max. A response time describes the time from calling an operation to getting a response from a service. An execution cost describes the total cost in terms of resources utilization. Reliability describes the rate in which services are executed correctly. A connection between services is achieved by exchanging message. A message describes datatype, name, unit and role. The business role gives the semantics of the corresponding parameter. It takes its value from a predefined taxonomy for business roles. In order to connect between two services the mapping between messages of two services must be completed. The functionalities provided by a service are accessible through operation invocations. We consider four operation modes. 'one-way', 'notification', 'solicit-response', and 'request-response'. This is described in detail, in Table 1.



**Table 1.** Operation mode ontology

| Operation mode   | Description  |
|------------------|--|
| one-way          | Input message, No output message. A pair with notification |
| Notification     | No Input message, Output message. A pair with one-way      |
| Solicit-response | Input – Output. A pair with request-response               |
| request-response | Output – Input. A pair with solicit-response               |

A composite service is treated as a logical service. This service is executed using transaction methods. We designed properties for a transaction. A state of transaction is defined with ‘commit’, ‘executing’, ‘compensate’, and ‘abort’ properties. An activity is defined with ‘started’, ‘completed’, ‘running’, and ‘blocked’.

### Service Composability

In this section, we describe how the framework can decide whether two services can be composed together. A composition of two services is achieved by an operation of one service call to an operation of another service. An calling of operation call is achieved by a sending a message. In order to decide whether two services can be composed, message compatibility and operation compatibility are verified.

**Message Compatibility :** Interoperation is achieved by exchanging messages. A message could be composed of parameters, having specific datatypes. A sending parameter must be interoperable with a receiving parameter. Every parameter would have well-defined semantics according to that taxonomy. We consider two primary data-type-compatibility methods: direct and indirect compatibility. Two parameters are directly compatible if they have the same data type. A parameter  $p$  is indirectly compatible with a  $q$  if the type of  $p$  is derived from the type  $q$ . We extend the notion of data type compatibility to messages as follows: A message  $M$  is a datatype compatible with a message  $N$  if every parameter of  $M$  is directly or indirectly compatible with a parameter of  $N$ . Note that not all parameters of  $N$  need to be mapped to the parameters of  $N$ .

**Operation Compatibility :** In order to be compatible with other operations, these operations have a “dual” mode. As described in Table 1, if an operation mode of one service is “one-way” then the other operation mode is “notification”. Also, two operations have the same purpose properties and category properties.

### Structure of Service Manager

Service discovery manager is composed by service composer, request handler, service finder and service register. Request Handler is a interface with services. Figure 4 describes service discovery manager.

Service finder can service reference which is stored in service registry by service query from services. Service register manages service registry which stores service description. Service composer is more complex component. This is composed by plan generate, plan selection, QoS evaluator, Agent factory and Agent pool. Service composer can generate service composition plan and agent which manage a execution of composition plan. Service wants other service with service request, service finder match service request with service description in service registry. If service is found,

service reference is passed to service. If service is not found, this request is passed to plan generator. Plan generator makes a composition plan of service using service finder. Plan selector choose a best composition plan among a few composition plans and make an agent which manage service execution of service composition plan. This agent is registered in service register as a virtual service.

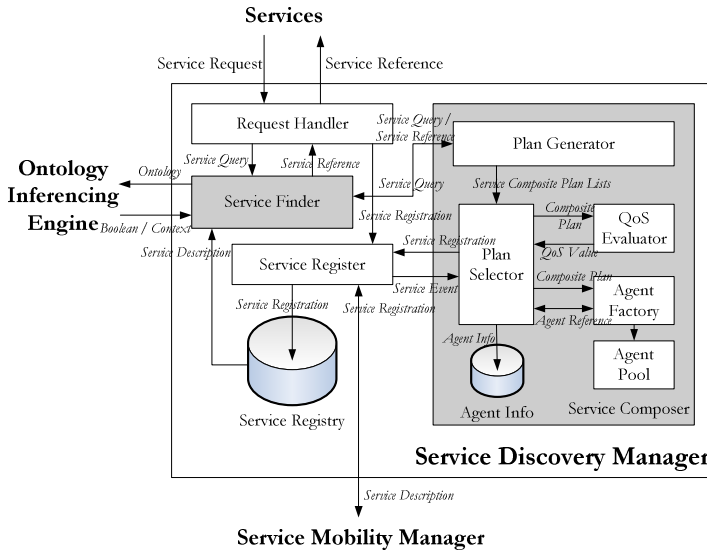


Fig. 4. Service Discovery Manager

## 4 Experiment

Table 2. displays the overall software configuration used for implementing the OSGi-based mobile agent management system proposed in this paper. During SOAP transmission from Knopflerfish using the Apache Axis, a minor bug was found in Java 1.5 or later, so the version 1.4.2 was used.

Table 2. The software configuration

| Software configuration      | Description          |
|-----------------------------|----------------------|
| SOAP                        | Apache Axis 1.2      |
| Java Binding                | Castor 0.97          |
| OSGi Framework              | Knopflerfish 1.3.3   |
| Ontology Inferencing Engine | Jena2                |
| Java Virtual Machine        | Java 1.4.2           |
| Operating System            | Microsoft Windows XP |

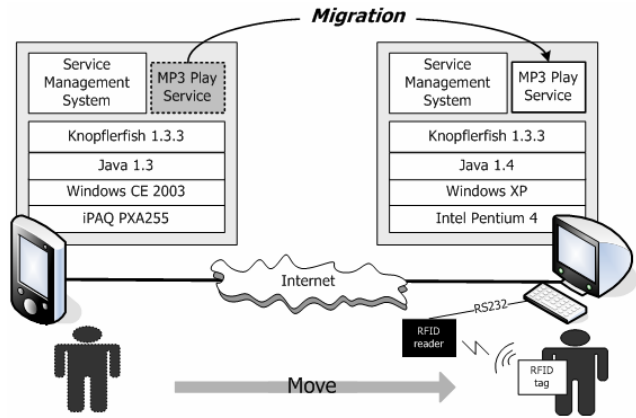


Fig. 5. The mobility of the MPlayer bundle

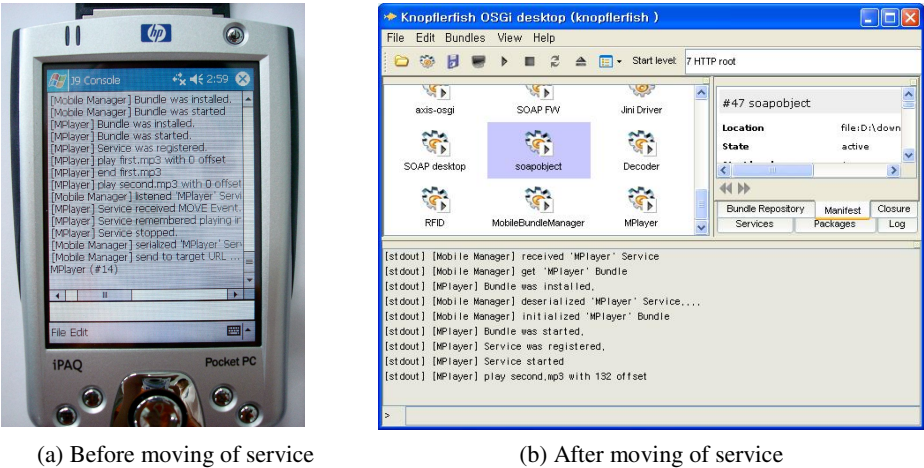


Fig. 6. The mobility of the MPlayer service

In order to test the service management system proposed in this paper, an MPlayer bundle was developed for playing MP3 music files. JVM and OSGi service management system bundles were installed in a PDA and PC, respectively as an experiment environment as shown in Figure 6.

When a user listening to music from the MPlayer installed in the PDA moves into the space where the PC is located, the MPlayer service was moved to the PC and the file was resumed from the point where it had been playing from the PDA, providing a continuous MPlayer service to the user.

Fig. 6 (a) displays the MPlayer in use from the PDA prior to service mobility. The state data serialized during mobility is the offset information regarding the music play list and the music file currently being played. The MPlayer does not have a GUI, and it is a bundle that plays the mp3 play list through a simple configuration file.

Fig. 6 (b) is the result screen after bundle mobility. The MPlayer bundle is automatically downloaded and installed using the bundle's class loading function, the service is initialized with the music play list offset data, and the music is played. And a description of this service is moved and registered in service registry. Implementation of the prototype displayed that the OSGi-based service management system proposed by this paper can operate as intended without much problem.

## 5 Conclusion

Service management system is inevitable in order to provide object mobility among OSGi frameworks and service discovery based on service semantic description constituting the ubiquitous computing environment such as the home network. This paper proposed a bundle in the form of a mobile service that can be autonomously executed in the OSGi framework, for which a mobile service lifecycle and a service mobility manager were designed and implemented for managing mobility. Also, it presents service discovery manager for service matching and composition based on service semantic description. The designed service management system was implemented in a bundle format to operate in the OSGi framework, and it also allowed dynamic management of autonomous services to provide mobility in a more efficient manner. In order to provide intelligent services in the future, there should be research efforts regarding OSGi-based situation recognition frameworks using the mobile service technology as well as security considerations for the mobile agent.

**Acknowledgments.** This work was supported by an INHA UNIVERSITY Research Grant.

## References

1. Open Services Gateway Initiative. <http://www.osgi.org>
2. D. Marples and P. Kriens, "The Open Services Gateway Initiative: An Introductory Overview," IEEE Communications Magazine, Vol. 39, No. 12, pp.110-114, December 2001
3. C. Lee, D. Nordstedt, and S. Helal, "Enabling Smart Spaces with OSGi," IEEE Pervasive Computing, Vol. 2, Issue 3, pp.89-94, July\_Sept. 2003
4. P. Dobrev, D. Famolari, C. Kurzke, and B. A. Miller, "Device and Service Discovery in Home Networks with OSGi," IEEE Communications Magazine, Vol. 40, Issue 8, pp.86-92, August 2002
5. K. Kang and J. Lee, "Implementation of Management Agents for an OSGi-based Residential Gateway," The 6th International Conference on Advanced Communication Technology, Vol. 2, pp.1103-1107, 2004
6. F. Yang, "Design and Implement of the Home Networking Service Agent Federation Using Open Service Gateway," International Conference on Integration of Knowledge Intensive Multi-Agent Systems, pp.628-633, Sept.\_Oct. 2003
7. H. Zhang, F. Wang, and Y. Ai, "An OSGi and Agent Based Control System Architecture for Smart Home," Proceedings of IEEE Networking, Sensing, and Control, pp.13-18, March 2005
8. Knopflerfish. <http://www.knopflerfish.org>

9. K. Chen and L. Gong, Programming Open Service Gateways with Java Embedded Server<sup>TM</sup> Technology, Addison Wesley, 2001
10. L. Gong, "A Software Architecture for Open Service Gateways," IEEE Internet Computing, Vol. 5, Issue 1, pp.64-70, Jan.-Feb. 2001
11. The DAML Service Coalition., OWL-S:Semantic Markup for Web Service, <http://www.daml.org/services/dalm-s>

# Creating Consistent Diagnoses List for Developmental Disorders Using UMLS

Nuaman Asbeh<sup>1</sup>, Mor Peleg<sup>2</sup>, Mitchell Schertz<sup>3</sup>, and Tsvi Kuflik<sup>2</sup>

<sup>1</sup> Department of Statistics, University of Haifa, Haifa, Israel, 31905  
nuamana@yahoo.com

<sup>2</sup> Department of Management Information Systems, University of Haifa, Haifa, Israel, 31905  
{morpeleg, tsvikak}@mis.haifa.ac.il

<sup>3</sup> Institute for Child Development, Kupat Holim Meuhedet, Central Region,  
Herzeliya, Israel 46782  
mitch\_s@meuhedet.co.il

**Abstract.** In the field of developmental disorders, there is no commonly accepted medical vocabulary. Vocabularies, such as ICD-10, are unsatisfying to clinicians, who try to create their own diagnostic lists. This results in inconsistency in the terms used in clinical practice. When attempting to apply automatic computational methods on patients' data, the need for common consistent diagnoses list arises. To this end, we mapped a set of different diagnoses used in clinical practice to UMLS—a well-known Unified Medical Language System that organizes and unifies over 100 vocabularies. Diagnoses that were defined by different terms and were mapped to one concept at UMLS were joined as synonyms. Concepts that were not found at UMLS were mapped to the closest concept found. We found that SNOMED-CT is the most comprehensive vocabulary (85.7% term coverage) in UMLS. We propose a framework that, when applied on inconsistent manually constructed set of diagnoses, leads to minimal and consistent set of diagnostic terms.

## 1 Introduction

High level of co-morbidity has long been recognized in childhood developmental disorders; children who have been diagnosed with one developmental disorder are very likely to meet diagnostic criteria for some other developmental disorder. This symptomatic overlap between different developmental disorders has led to a high rate of misdiagnosis. Clinical vocabularies, such as DSM-IV (American Psychiatric Association 1994) and ICD-10 (World Health Organization 1992), classify developmentally and psychiatrically disturbed children in terms of their emotional symptoms and behaviors while ignoring the underlying mechanisms, resulting in multiple overlapping diagnoses that may refer to the same underlying mechanism. Thus, these vocabularies are deeply unsatisfying to many clinicians, who try to create their own diagnostic lists. This results in inconsistency in the diagnostic terms used in clinical practice.

Developmental disorder diagnoses may be grouped into super diagnoses groups according to their underlying mechanisms that cause common findings that are shared by the disorders. Evidence for the existence of such groups has been shown [1]. To

this end, we are applying clustering analysis on patient data, to group developmental disorder diagnoses into clusters of super diagnoses. However, for the clustering results to be valid, the set of diagnoses (features) upon which the clustering methods are used, need to be defined clearly and consistently used by all the clinicians who diagnose pediatric patients. Therefore, we decided to develop a developmental-disorder ontology that is satisfying and acceptable to practitioners in the field and that will ensure that all practitioners who give diagnoses to children are using the same agreed-upon terminology and interpret the terms in a consistent way.

An ontology defines a common vocabulary for researchers who need to share information in a domain. It includes machine-interpretable definitions of basic concepts in the domain and relations among them [2]. Some of the reasons for developing ontologies are: (1) to share common understanding of the structure of the information among people or software agents, (2) to enable reuse of domain knowledge, (3) to make domain assumptions explicit.

## 2 Methods

We used the protégé-2000 [2] knowledge modeling tool to create the ontology of developmental disorders. We started from a set of diagnoses that are used in clinical practice at the Institute for Child Development, Kupat Holim Meuhedet, Central Region, Herzeliya. We mapped these diagnoses to medical concepts defined in the UMLS (Unified Medical Language System [3]) Metathesaurus—a well-known vocabulary system that organizes and unifies other vocabularies. Diagnoses that were defined by different terms and were mapped to the same concept at UMLS were joined together as synonyms. The clinician expert on our team (MS) mapped concepts that were not found in the UMLS Metathesaurus to the closest concept in UMLS (e.g., the term “sleep disorder – discontinuous” was mapped to the term “Sleep Initiation and Maintenance Disorders”). We considered each concept that was not found in UMLS, but appeared in the diagnosis set of the clinicians, as a candidate for entering the ontology's hierarchy as subclasses of existing classes. We inserted candidates as subclasses if they were specializations of diagnoses, based on severity (e.g., mild developmental delay), certainty of the diagnosis (e.g., ADHD-preschool is a specialization of ADHD since 50% of preschoolers diagnosed with ADHD are no longer diagnosed with it at school age), or causality (e.g., developmental memory disorders as a subtype of memory disorders). We would not add concepts to the ontology if the missing concepts are related to local environments or genetic influences because this would compromise the acceptance of the ontology by wider community of clinicians, worldwide.

We structured each medical concept in the ontology by defining slots taken from the way by which terms are defined in the UMLS Metathesaurus. These slots include name, synonyms, semantic types, textual definitions, concept\_identifier, and the source vocabularies in which it is found. We defined terms that were not found in UMLS using their original name and a unique concept\_identifier.

In order to construct the ontological links (hierarchical and non-hierarchical), we found the minimal coverage set (MCS)-the minimal set of vocabularies that cover all the concepts found in UMLS, and checked which vocabulary covers most of the

concepts in the ontology (SNOMED-CT, as discussed in section 3). The taxonomy of this vocabulary served as a basis for the ontology's hierarchy, which we call the Main Tree (MT). To create the MT, we manually entered terms to the ontology to ensure using only the MCS set; we entered all the ancestors of the terms found in the selected vocabulary. We then added the terms that were in MCS but not at the vocabulary used to create the MT: we followed the ancestral hierarchies of such a term, as it exists in the vocabularies in MCS-MT, searching for an ancestor that exists at MT. We added the ancestors up to the mutual ancestor to MT. In this way, we were able to link few more concepts into MT. The rest of the concepts remained in separate hierarchies, taken from their source vocabularies.

3 Results

Mapping the terms to UMLS minimized the set of 179 inconsistent diagnoses to an equivalent set of 88 consistent diagnoses, by inserting 91 diagnoses as synonyms of the other 88. The 88 diagnoses were entered to the ontology. Seventy seven of the 88 terms were found in UMLS (87.5%), and 11 terms were not found in UMLS but are subtypes of terms found in UMLS (12.5%), as explained in the Methods. The diagnosis list was approved by the clinician expert on our team (MS) to ensure clinical validity of the list.

**Table 1.** Percentage of diagnoses covered by clinical vocabularies found in UMLS

| Vocabulary                       | Number of diagnoses covered | %diagnoses covered |
|----------------------------------|-----------------------------|--------------------|
| SNOMED-CT                        | 66                          | 85.7               |
| SNOMED-Intl-1998                 | 59                          | 76.6               |
| MedRA                            | 56                          | 72.7               |
| ICD-9-CM, ICD-10, &DSM-IV        | 51                          | 66.2               |
| MESH                             | 43                          | 55.8               |
| Alcohol and Other Drug Thesaurus | 37                          | 48.0               |
| ICD-10                           | 34                          | 44.2               |
| ICD-9-CM                         | 31                          | 40.3               |
| DSM-IV                           | 28                          | 36.4               |
| Clinical Problem Statements      | 25                          | 32.5               |
| ICPC2E-1998                      | 2                           | 2.6                |

We found that 18 vocabularies include terms from the diagnostic list and 7 of them are sufficient to create a MCS. The results showed that the SNOMED-CT vocabulary is the most comprehensive vocabulary, covering all but 11 concepts found in UMLS (85.7% coverage). We therefore used its taxonomy as the ontology's concept hierarchy and were able to integrate seven of the 11 concepts with SNOMED-CT's hierarchy since ancestors of these concepts were included in SNOMED-CT. The remaining four concepts were integrated as separate hierarchies originating from their source vocabularies. Table 1 shows the percentage of diagnoses that are covered by the MCS and by DSM-IV, ICD-9, and IDC-10, which are the vocabularies most used



in clinical practice in this field. The hierarchies of DSM-IV and ICD conformed with the MT of SNOMED-CT for 47 of the 51 overlapping terms.

## 4 Discussion

We reported a framework for taking an inconsistent manually constructed set of diagnoses and forming a consistent set of diagnostic terms that is based on established medical vocabularies. This standardization of the terms used at the diagnostic list brings at least two advantages (1) allowing knowledge sharing in the domain – the ontology may serve as a starting point for establishing an internationally agreed upon list of developmental disorder diagnoses and (2) allowing the application of computational methods on patients' data to discover new knowledge (e.g., we can now apply clustering techniques for finding patients with similar comorbidities and analyze the similarity).

The framework we proposed showed its potential to identify the most comprehensive vocabulary (SNOMED-CT in our study) of the clinical vocabularies that cover a medical sub-domain, suggesting that it may serve as the main vocabulary used in the clinical sub-domain. In clinical practice, clinicians in Israel do not use SNOMED-CT, and instead use other vocabularies as the basis for their diagnoses lists: DSM-IV, ICD-10, or ICD-9-CM. While each of these vocabularies alone is not comprehensive enough, combining the three of them covers 51 (66.2%) of the terms found in UMLS.

The ontology is not finalized and needs to be further validated by more experts. Special care will be paid to the 11 diagnostic terms that were not found in UMLS. In some cases, they represent new knowledge that has not yet been added to existing vocabularies, and clear definitions must be established to allow for consistency in diagnoses made by different clinicians.

## References

1. Gillberg, C.: Deficits in attention, motor control and perception, and other syndromes attributed to minimal brain dysfunction. In Gillberg C, (ed).: Clinical child neuropsychiatry. Cambridge University Press, Cambridge, UK (1995) 138-172.
2. Noy N., McGuinness D.L.: Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory Technical Report No. KSL-01-05 and Stanford Medical Informatics Technical Report # SMI-2001-0880; (2001).
3. Humpreys B., Lindberg D., Schoolman H., Barnett G. The Unified Medical Language: An Informatics Research Collaboration. J Am Med Inform Assoc. 5:1 (1998) 1-11

# Enhancing Domain Engineering with Aspect-Orientation

Iris Reinhartz-Berger<sup>1</sup> and Alex Gold<sup>2</sup>

<sup>1</sup> Department of Management Information Systems,  
University of Haifa, Haifa 31905, Israel  
[iris@mis.haifa.ac.il](mailto:iris@mis.haifa.ac.il)

<sup>2</sup> Department of Computer Science,  
University of Haifa, Haifa 31905, Israel  
[alexgold@013.net.il](mailto:alexgold@013.net.il)

Domain engineering is the process of creating common knowledge applicable for a family of similar systems. Its activities include identifying the domain terminology, capturing the possible variations within the domain, constructing adaptable designs, and defining mechanisms for instantiating particular systems in the domain. The artifacts of these activities are domain models, domain designs, domain-specific languages, code generators, and reusable components [2]. In our work we focus on domain modeling.

Although domain engineering is a well established field with many techniques that have been proposed over the years, the questions of what can be defined as a domain and how domains are related have not been researched enough. In particular these questions should be addressed in the context of analyzing the tradeoffs between defining wide domains that are general enough to support various applications and defining narrower domains that capture the domain knowledge in more details. We believe that there is no single (best) way to answer these questions; hence, we suggest supporting different levels of details within domain engineering. Two examples for using such levels are given next.

**Dividing Domains into Sub-domains.** Sometimes, the defined domains are too wide and the knowledge that can be acquired about them is too general. Dividing these domains into sub-domains that are more specific, but are still quite wide to include large numbers of applications, can help refining the (sub-)domain models. For example, consider the case of e-commerce applications. All e-commerce applications should handle a kernel of activities, such as browsing a catalog, issuing requests, paying, processing delivery, and confirming shipment. However, the wide spectrum of e-commerce applications includes auction sites on one hand and e-stores on the other hand. While e-stores offer their goods (products or services) in fix prices, auction sites enable selling the same product from the same merchant at different prices to different customers. Treating this variety as a single domain affects the level of details to which we can reach. It will be helpful if we could specify the general concepts in an e-commerce domain model and then refine this model, by specialization, aggregation, etc., to separately support auction sites and e-stores.

**Weaving New Aspects to Existing Domains.** Sometimes, new aspects or concerns that were not treated when originally analyzing, designing, and implementing the domain should be added [1], for example, enhancing e-commerce applications with

security and privacy policies. When modeling these policies separately from the core e-commerce model, they can be woven to other domains as well.

In the software engineering world, these problems get answers in the aspect-oriented discipline. Hence, we suggest integrating aspect-orientation also with domain engineering. In this work, we incorporate a specific aspect-oriented technique with the Application-based Domain Modeling (ADOM) approach in order to support the definition of relationships between domains and to explore the domain specialization process. ADOM [4] is a domain engineering method, which perceives that applications and domains require similar facilities for their modeling, design, and development, thus it enables specifying and constructing domain artifacts with regular application engineering tools and techniques. The application models are then validated against the relevant domain models.

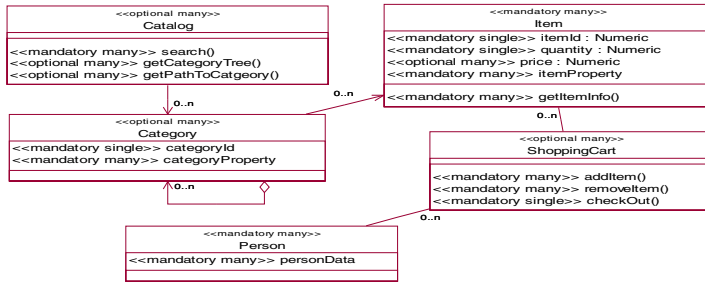
The general architecture of ADOM consists of three layers: application, domain, and language. The *application layer* includes models of particular applications, e.g., Amazon and e-Bay web sites. The *language layer* includes metamodels of modeling languages, such as UML. The intermediate *domain layer* consists of specifications of various domains, e.g., e-commerce applications, auction sites, and e-stores. In addition, the ADOM approach enforces constraints between the different layers.

Although ADOM relies on a general, language-independent architecture, we applied it to the standard object-oriented modeling language, UML. In order to support the variability of the domain applications, ADOM utilizes UML stereotype mechanism in both domain and application layers. In the domain layer, "multiplicity" stereotypes are used to represent how many times a model element can appear in a particular application model. In the application layer, the different elements are stereotyped and classified according to the domain elements. An application model element must preserve the relations and constraints of its stereotypes in the relevant domain models.

In order to improve the reusability, maintainability, and comprehensibility of the models in the domain layer of ADOM, we apply aspect-orientation to ADOM. The aspect-oriented approach [1] provides systematic means for the identification, modularization, representation, and composition of crosscutting concerns. The Early Aspects initiative [3] focuses on managing crosscutting properties at the early development stages of requirements engineering and architecture design. Several works have been carried out to percolate the notion of an aspect to early development stages. Most of them rely on UML and introduce sets of stereotypes or extend the UML metamodel to support the aspect-oriented concepts. In this work, we decide to focus on the Aspect-Oriented Design Modeling (AODM) technique [5] because of the following reasons. First, it uses standard extension mechanisms rather than extending the metamodel. Second, it supports the specification of both behavioral and structural crosscutting concerns, enriching the expressiveness of the resultant models. Third, this approach specifies how an aspect-oriented design model may be transformed into an ordinary UML model, enabling an easier integration with ADOM.

As an example of how ADOM works together with AODM, consider the e-commerce application domain and the auction site sub-domain. Figure 1 is part of an e-commerce domain model in ADOM. This part depicts in terms of a class diagram the structure and relations of five major concepts: Catalog, Category, Item, Shopping Cart, and Person. As specified by the multiplicity stereotypes, each application in the domain may have zero or more (optional many) catalogs organized

in categories and zero or more types of Shopping Carts. In addition, each application must have at least one (mandatory many) type of items and at least one type of persons (e.g., customers). The model also specifies the internal structure of each concept. An application class classified as an Item, for example, must have a numeric identity, a numeric field representing its quantity in stock, one or more numeric fields representing its prices (e.g., to different types of customers), and at least one additional property (e.g., name, color, size, etc.). In addition, item classes must have at least one method for retrieving the item details (in different sections).



**Fig. 1.** A part of an e-commerce domain model in ADOM

Auction sites, which are e-commerce applications, mainly differ from other applications in the domain due to their bidding processes that help determining item prices. There are different types of auctions, such as First Price Auctions, English Auctions, Dutch Auctions, and others. They all introduce different bidding algorithms. Some of the changes that auction sites make to the original e-commerce domain model (Fig. 1) are adding two new concepts, Auction (along its bidding algorithm) and Proposal. These concepts should be connected to existing concepts, such as Person and Item. The ADOM model of this auction site aspect (Fig. 2a) includes the two (new) classes that should be introduced to the original model (Fig. 1). The expected structure (attributes and relations) and behavior (methods) of each class are also specified. Fig. 2b provides the guidelines (in AODM) for weaving the aspect model from Fig. 2a into the original model (Fig. 1). In particular, the model specifies that the (new) Auction class should be connected to the (existing) Item class from the original model and that the (new) Proposal class should be connected to the (existing) Person class. After weaving the auction site aspect model specified in Fig. 2 with the e-commerce domain model specified in Fig. 1, the woven auction site domain model presented in Figure 3 is achieved.

The challenges that we have still to face are: (1) Extending the approach to support different structural and behavioral views of the domain, e.g., use case, class, and interaction diagrams. In particular, we will focus on the new challenges that the domain engineering field raises over regular software or application engineering (e.g., the requirement to capture the possible variations within the domain), (2) Enabling addition, modification, and deletion (override) of elements and constraints from the general domain by the specific sub-domain. (3) Verifying and evaluating the approach in terms of expressiveness, clearness, easiness of use, etc.

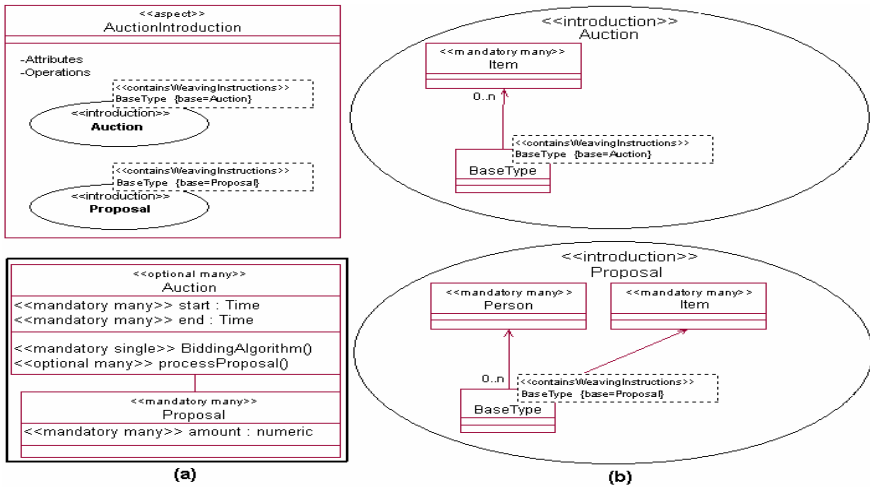


Fig. 2. An auction site aspect model in (a) ADOM and (b) AODM

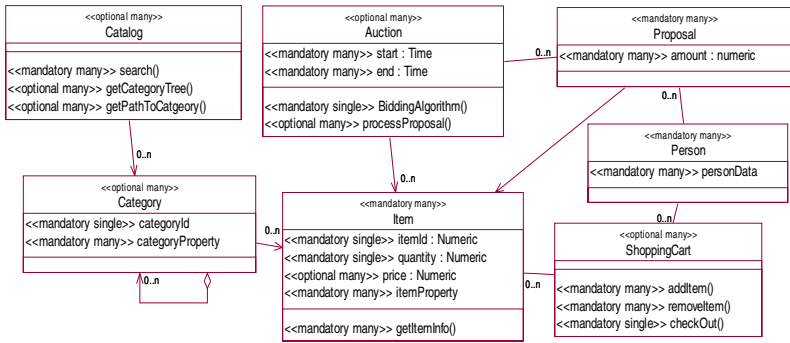


Fig. 3. The woven auction site domain model in ADOM

## References

1. Aspect-Oriented Software Development Community & Conference web site, <http://aosd.net/>
2. Carnegie, M. "Domain Engineering: A Model-Based Approach", Software Engineering Institute, <http://www.sei.cmu.edu/domain-engineering/>, 2004.
3. Early Aspects: Aspects-Oriented Requirements Engineering and Architecture Design web site, <http://www.early-aspects.net/>
4. Reinhartz-Berger, I. and Sturm, A. "Behavioral Domain Analysis – The Application-based Domain Modeling Approach", the 7th International Conference on the Unified Modeling Language (UML'2004), Lecture Notes in Computer Science 3273, pp. 410-424, 2004
5. Stein, D., Hanenberg, S., Unland, R., "A UML-based Aspect-Oriented Design Notation For AspectJ". Proc. of 1<sup>st</sup> International Conference on Aspect-Oriented Software Development (AOSD'2002), ACM, pp. 106-112, 2002.

# An Agent-Oriented Approach to Semantic Web Services

In-Cheol Kim

Department of Computer Science, Kyonggi University  
Suwon-si, Kyonggi-do, 442-760, South Korea  
kic@kyonggi.ac.kr

**Abstract.** The current architectures for semantic web service do not provide with integrated functionality of automated composition, dynamic binding, and invocation. Openness and dynamics of the Web environment limits the usage of previous approaches based upon the traditional AI planning techniques. This paper introduces a BDI agent system for semantic web service composition and invocation. Through some tests on healthcare web services, we found our agent-oriented approach has the potential enough to improve robustness and flexibility of semantic web services.

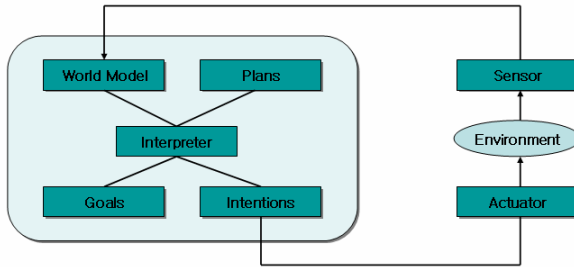
## 1 Introduction

Semantic web services augment web services with rich formal descriptions of their capabilities, thus facilitating automated discovery, composition, dynamic binding, and invocation of services within an open environment. Semantic web services, therefore, have the potential to change the way knowledge and business services are consumed and provided on the Web. Enabling semantic web services needs the infrastructure such as standard service ontology and architecture. The service ontology, such as OWL-S, aggregates all concept models related to the description of a semantic web service, and constitutes the knowledge-level model of the information describing and supporting the usage of the service. Currently several tools supporting OWL-S have been developed: OWL-S Editor, Matchmaker, Broker, Composer, and Virtual Machine [3]. However, the current semantic web service architectures do not provide with integrated functionality of composition, dynamic binding, and invocation [1]. Although there are some efforts for automated semantic web service composition, most of them are based upon the traditional AI planning techniques. Due to openness and dynamics of the Web environment, the web services composed through off-line planning are subject to fail. In order to overcome the drawback, this paper introduces a BDI agent system for semantic web service composition and invocation. Using some examples of healthcare web services, we test the feasibility of our approach.

## 2 BDI Agent Architecture

The most commonly used architecture for software agents is the *Belief-Desire-Intention* (BDI) model. Fig. 1 shows a PRS-based BDI architecture called JAM [2].

Each JAM agent is composed of five primary components: a *world model*, a *goal set*, a *plan library*, an *interpreter*, and an *intention structure*. The world model is a database that represents the beliefs of the agent. The goal set is a set of goals that the agent has to achieve. The plan library is a collection of plans that the agent can use to achieve its goals. The interpreter is the agent's "brain" that reasons about what the agent should do and when and how to do it. The intention structure is an internal model of the agent's current goals and keeps track of the commitment to, and progress on, accomplishment of those goals.



**Fig. 1.** JAM: A BDI Agent Architecture

The BDI architecture integrates traditional goal-directed reasoning and reactive behavior. Because most traditional deliberative planning systems formulate an entire course of action before starting execution of a plan, these systems are brittle to the extent that features of the world or consequences of actions might be uncertain. In contrast, the BDI architecture continuously tests its decisions against its changing knowledge about the world, and can redirect the choices of actions dynamically while remaining purposeful to the extent of the unexpected changes to the environment.

### 3 SWEEP II System

SWEEP II is a BDI agent system supporting automated semantic web service composition and invocation. As shown in Fig. 2, the core component of SWEEP II is the *JAM BDI engine*. Additionally, SWEEP II includes several main components such as *service description manager*, *OWL-S2JAM converter*, *ontology manager*, *reasoner*, *query processor*, *mediator*, *task manager*, *web service invoker*. The service description manager retrieves and stores OWL-S descriptions from the semantic web service repository. The OWL-S2JAM converter transforms the retrieved OWL-S descriptions into the corresponding JAM primitive plans.

Based upon the domain ontologies managed by the ontology manager, the reasoner provides reasoning services for service matching, validation, and mediation. The query processor takes a request from the user. With the help of the mediator, it generates a task to be accomplished by JAM. In order to accomplish the given task, JAM

reactively elaborates a complex plan from primitive plans considering its goals and the world model. According to the intentions of JAM, the web service invoker calls the atomic web services in order. Fig. 3 illustrates the mapping relationship from an OWL-S service description to the corresponding JAM plan. Outputs and effects of the OWL-S process are mapped to goals and effects of the JAM plan. Inputs and preconditions of the OWL-S process are mapped to subgoals and context of the JAM plan. The grounding WSDL operations of the OWL-S service are mapped to the external functions of the JAM plan. Many ontological concepts and properties in the OWL-S description are converted into the corresponding relations within the JAM world model. Due to features of the embedded JAM engine, the SWEEP II system can adapt its decisions on web service composition and invocation against the changing Web environment. In this sense, the SWEEP II system realizes *dynamic composition* of semantic web services.

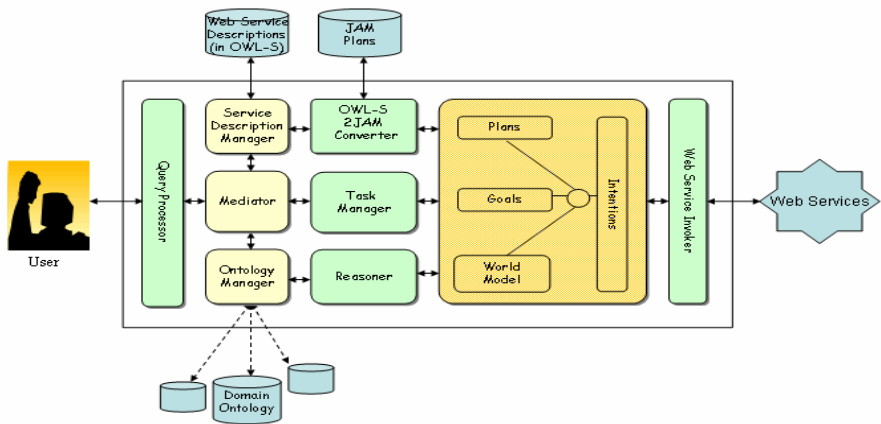


Fig. 2. Architecture of the SWEEP II System

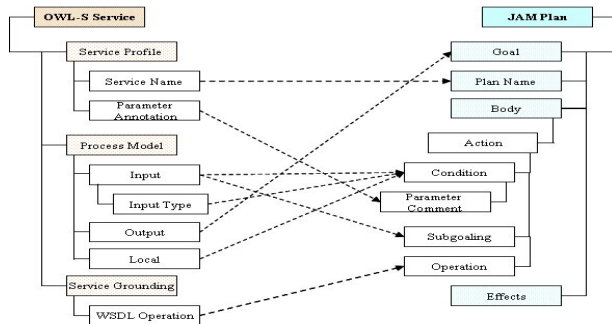
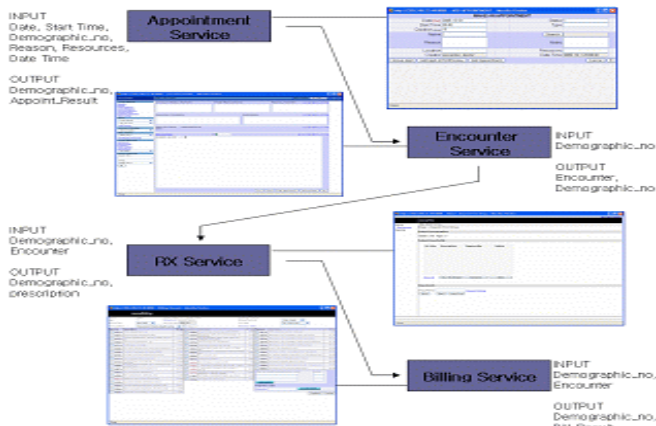


Fig. 3. OWL-S to JAM Plan Mapping





**Fig. 4.** Healthcare Web Services

In order to test the feasibility of our SWEEP II system, we developed some examples of healthcare web services shown in Fig. 4. They are appointment service, encounter service, RX service, and billing service for patients. These are typically common services provided by many hospital information systems (HIS). For our purpose, however, we implemented them as OWL-S semantic web services. Through some tests on these semantic web services, we found that our SWEEP II system shows high robustness against unexpected failures and delays of web services.

## 4 Conclusions

We introduced an agent-oriented approach to semantic web service composition and invocation. Through some tests on healthcare web services, we found that our approach has the potential enough to improve robustness and flexibility of semantic web services.

## References

1. Cabral, L., Domingue, J., Motta, E., Payne, T., Hakimpour, F.: Approaches to Semantic Web Services: An Overview and Comparisons. Proceedings of the 1st European Semantic Web Symposium (ESWS2004), Springer-Verlag (2004) 225-239
2. Marcus, J.: JAM: A BDI-theoretic Mobile Agent Architecture. Proceedings of the 3<sup>rd</sup> International Conference on Autonomous Agents (Agents'99) (1999) 236-243
3. Paolucci, M., Sycara K.: Autonomous Semantic Web Services. IEEE Internet Computing, Vol.7(5). (2003) 34-41

# Biometrics Authenticated Key Agreement Scheme

Eun-Jun Yoon and Kee-Young Yoo\*

Department of Computer Engineering, Kyungpook National University,  
Daegu 702-701, Republic of Korea  
ejyoon@infosec.knu.ac.kr, yook@knu.ac.kr

**Abstract.** The current paper presents an efficient and secure biometrics authenticated key agreement scheme based on one-way hash function. In the proposed scheme, after a user passes user authentication check of a remote system, they agree on a session key for protecting their subsequent communications.

**Keyword:** User authentication, Smart card, Fingerprint, Biometrics.

## 1 Introduction

User authentication is an important part of security, along with confidentiality and integrity, for systems like the Internet that offer remote access over untrustworthy networks. In 2002, Lee et al. [1] proposed a biometrics remote user authentication scheme using smart cards based on the ElGamal's public key cryptosystem [2] with two secret keys. Since a different map of minutiae can be made when the input device takes a smart card owner's fingerprint, Lee et al.'s scheme can generate a one-time random number for the ElGamal's public key cryptosystem by using the map of minutiae. However, in 2004, Lin-Lai [3] pointed out that Lee et al.'s scheme does not withstand masquerade attack by using two secret keys and fingerprint verification. Furthermore, they proposed an improved scheme to enhance security, which allows users to choose and change their password conveniently. Unlike Lee et al.'s scheme, based on ElGamal's cryptosystem and the fingerprint verification, Lin-Lai's scheme needs only to maintain one secret key, without verification tables such as password and identity tables. However, both Lee et al.'s and Lin-Lai's schemes still required high communication and computation costs such as modular exponential operations. Therefore, the current paper presents an improved scheme based on a one-way hash function with better security strength and efficiency. Unlike the preceding schemes, in the proposed scheme, after a user passes user authentication check of a remote system, they agree on a session key to protect their subsequent communications. Thereby, the transmitted messages between the user and the remote system can be kept secret when the user uses a service of the remote system.

---

\* Corresponding author: Tel.: +82-53-950-5553; Fax: +82-53-957-4846

## 2 Biometrics Authenticated Key Agreement Scheme

This section proposes a biometrics authenticated key agreement scheme. There are three phases in the proposed scheme: a registration phase, a login phase, and a key agreement phase. Additionally, we propose a secure password change scheme that does not use help the remote system.

### Registration Phase

- R.1 Before accessing a remote system, a new user  $U_i$  selects his or her identity  $ID_i$ , password  $PW_i$  and a random number  $N$  freely. Then, he or she personally imprints his fingerprint on the input device, computes  $h(PW_i, N)$ , and sends it with  $ID_i$  to the remote system  $S$ , where  $h(\cdot)$  denotes a one-way hash function.
- R.2 If it is  $U_i$ 's initial registration,  $S$  creates an entry for  $U_i$  in the account database and stores  $n = 0$  in this entry. Otherwise,  $S$  sets  $n = n + 1$  in the existing entry for  $U_i$ . Next,  $S$  computes  $K = h(X_S, ID_i, n)$  and  $R = K \oplus h(PW_i, N)$ , where  $X_S$  denotes the secret key kept securely in the remote system  $S$  and  $\oplus$  denotes a bit-wise exclusive-or operation, and then writes  $\{R, h(\cdot)\}$  into  $U_i$ 's smart card and releases it to  $U_i$  through a secure channel.
- R.3  $U_i$  enters  $N$  into his smart card.

### Login Phase

- L1. Whenever a user  $U_i$  wants to login,  $U_i$  inserts his or her smart card into the card reader, keys in  $ID_i$  and  $PW_i$ , and imprints his or her fingerprint on the fingerprint input device. If the fingerprint of  $U_i$  is not successfully verified, the login process is terminated.
- L2.  $U_i$ 's smart card computes  $h(PW_i, N)$  by using stored  $N$  and extracts  $K = R \oplus h(PW_i, N)$ . Then  $U_i$ 's smart card generates a random number  $r$  using coordinates of the minutia of the input fingerprint, and then sends his or her login request message  $ID_i$  and  $r$  to  $S$ .

### Key Agreement Phase

- K.1 Upon receiving  $U_i$ 's login request message,  $S$  checks the validity of  $ID_i$ . If  $ID_i$  is not a valid identity,  $S$  rejects  $U_i$ 's login request. Otherwise,  $S$  retrieves  $n$  from the account database of  $U_i$  and then computes  $K' = h(X_S, ID_i, n)$ . Then,  $S$  chooses a random nonce  $r'$  and computes  $h(K, r, r')$ . Finally,  $S$  sends its to  $U_i$ .
- K.2 Upon receiving  $r'$  and  $h(K, r, r')$ ,  $U_i$ 's smart card verifies  $h(K, r, r')$  contains  $K$  to authenticate  $S$ . If the result is positive,  $U_i$  can ensure that  $S$  is legal. Finally, with  $r'$ ,  $U_i$ 's smart card computes  $h(K, r', r)$  as his or her authentication token and sends it to  $S$ .
- K.3 Upon receiving  $h(K, r', r)$ ,  $S$  computes  $h(K', r', r)$  and verifies  $h(K, r', r)$  contains  $K$  to authenticate  $U_i$ . If the result is positive,  $S$  can ensure that  $U_i$  is legal. After mutual authentication between  $U_i$  and  $S$ ,  $SK = h(ID_i, K, r, r')$  is used as the session key for protecting their subsequent communications.

### Password Change Scheme

- P.1 Whenever  $U_i$  decides to change the old password  $PW_i$  to a new password  $PW'_i$ ,  $U_i$  imprints his or her fingerprint on the fingerprint input device.
- P.2 If the fingerprint of  $U_i$  is not successfully verified, the password change process is terminated. Otherwise,  $U_i$  inputs the old password  $PW_i$  and the new password  $PW'_i$ .
- P.3  $U_i$ 's smart card computes a new  $R' = R \oplus h(PW_i, N) \oplus h(PW'_i, N)$ , and then replaces the old  $R$  with the new  $R'$  on the smart card.

## 3 Security and Efficiency Analysis

This section analyzes the security and efficiency of the proposed scheme.

**Security Analysis:** (1) The proposed scheme can resist guessing attacks. Due to the fact that a one-way hash function is computationally difficult to invert, it is extremely hard for any attacker  $E$  to derive  $X_S$  from  $h(X_S, ID_i, n)$  and  $K$  from  $h(K, r, r')$  (or  $h(K, r', r)$ ). (2) The proposed scheme can resist replay attacks.  $E$  intercepts  $(ID_i, r)$  sent by  $U_i$  in Step (L.2) and uses it to impersonate  $U_i$  when sending the next login message. However, for a random challenge,  $r$  separately generated by  $U_i$  is different every time. As a result, the replay of  $U_i$ 's old login messages in Steps (L.2) and (K.2) are detected by  $S$  because  $E$  has no  $K$  with which to compute a correct  $h(K, r', r)$ . (3) The proposed scheme can resist impersonation attacks.  $E$  can attempt to modify a message  $(ID_i, r)$  into  $(ID_i, r^*)$  and send it to  $S$ , where  $r^*$  is a random nonce selected by  $E$ . However, such a modification will fail in Step (K.3) of the key agreement phase, because  $E$  has no way of obtaining  $K$  to compute the valid  $h(K, r', r^*)$ . If a masquerading  $S$  tries to cheat the requesting  $U_i$ , it has to prepare a valid message  $(r', h(K, r, r'))$ . However, this is not feasible, as there is no way to derive  $K$  to compute  $h(K, r, r')$ , due to the one-way property of a secure one-way hash function. (4) The proposed scheme can resist insider attacks. Since  $U_i$  registers to  $S$  by presenting  $(ID_i, h(PW_i, N))$  instead of  $(ID_i, PW_i)$ , unlike Lin-Lai's scheme, the insider of  $S$  cannot directly obtain  $PW_i$  without knowing random nonce  $N$ . (5) The proposed scheme is easily reparable. If  $U_i$  finds or suspects that his or her  $K$  has been compromised, he or she can select a new random number  $N'$  and new password  $PW'_i$ , and then compute  $h(PW'_i, N')$ . Next,  $U_i$  can re-register to  $S$  by using  $h(PW'_i, N')$ . Upon receiving  $U_i$ 's re-registration request,  $S$  will set  $n' = n + 1$  and compute  $K' = h(X_S, ID_i, n')$  and  $R' = K' \oplus h(PW'_i, N')$ . Next,  $S$  stores  $R'$  in  $U_i$ 's new smart card. After receiving the new smart card from  $S$  through a secure channel,  $U_i$  enters  $N'$  into it. From now on,  $U_i$  can securely login  $S$  by using the new smart card and new password  $PW'_i$ . Meanwhile, the compromised  $K$  has been revoked automatically; i.e., the login request of  $E$  who has obtained  $K$  will be rejected. (6) The proposed scheme provides mutual authentication. The proposed scheme uses the session key exchange algorithm to provide mutual authentication. Then, the key is explicitly authenticated by a mutual confirmation values  $h(K, r, r')$  and  $h(K, r', r)$ , respectively. (7) The proposed scheme provides session key security. The session key  $SK = h(ID_i, K, r, r')$  is

**Table 1.** Comparisons of computational costs

|  | Lin-Lai's scheme |               | Proposed scheme |         |
|--|------------------|---------------|-----------------|---------|
|  | User             | System        | User            | System  |
| Registration   | No               | 1e+1h+2xor    | 1h              | 2h+1xor |
| Login and Authentication (unilateral authentication) | 2e+2h+1m+3xor    | 2e+1h+1m+1xor | 1h+1xor         | 1h      |
| Login and Authentication (mutual authentication)     | N/A              | N/A           | 4h+1xor         | 4h      |
| Password change scheme                               | 2h+xor           |               | 2h+2xor         |         |

e: exponentiation operations; h: secure one-way hash operations;  
m: multiplication operations; xor: bitwise exclusive-or operations.

known only to  $U_i$  and  $S$  since the session key  $SK$  is protected by  $K$  and the secure one-way hash function. (8) The proposed scheme provides perfect forward secrecy. If  $E$  can get some used random values  $r$  and  $r'$  and the shared long-term secret key  $K$  or  $X_S$ ,  $E$  can compute the used session key  $SK$ . To remedy this problem, the Diffie-Hellman key exchange algorithm can be used to compute the session key. In this approach, we let  $r = g^a$  and  $r' = g^b$ , where  $a$  and  $b$  are random exponents chosen by  $U_i$  and  $S$  separately, and  $SK = h(ID_i, K, g^{ab})$ .

**Efficiency Analysis:** The computational costs of Lin-Lai's scheme [3] and the proposed scheme in registration, login, authentication, and change password phases are summarized in Table 1. In the registration, login, authentication, and change password phases, Lin-Lai's scheme requires a total of 5 times exponentiation operations, 6 times hash operations, 2 times multiplication operations and 10 times bitwise exclusive-or operations for unilateral authentication, but the proposed scheme requires only a total of 7 times hash operations and 4 times bitwise exclusive-or operations. For mutual authentication, the proposed scheme requires a total of 11 times hash operations and 2 times bitwise exclusive-or operations. It is clear that the proposed scheme is more efficient than Lin-Lai's scheme.

## 4 Conclusions

In the current paper, an enhancement to Lin-Lai's flexible biometrics remote user authentication scheme was proposed. The proposed scheme offers the same advantages as Lin-Lai's scheme and has the following additional merits: (1) The remote system spoofing attack is completely solved by providing mutual authentication. (2) The scheme provides session key agreement. (3) The computational costs are fewer than Lin-Lai's proposed scheme.

**Acknowledgements.** We would like to thank the anonymous reviewers for their helpful comments in improving our manuscript. This research was supported by the MIC (Ministry of Information and Communication), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Assessment).

## References

1. J. K. Lee, S. R. Ryu, K. Y. Yoo, Fingerprint-based remote user authentication scheme using smart cards, *Electronics Letters*, Vol. 38, No. 2, pp. 554-555, 2002.
2. T. ElGamal, A public-key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Transactions on Information Theory*, Vol. IT-31, No. 4, pp. 469-472, 1985.
3. C. H. Lin, Y. Y. Lai, A flexible biometrics remote user authentication scheme, *Computer standards & Interfaces*, Vol. 27, No. 1, pp. 19-23, 2004.

# The Third Query Paradigm

Mikhail Gilula

DataArk, CA 94404-2782, USA  
gilula@DataArk.net

**Abstract.** We aim to conceptualize a query paradigm for distributed semantically heterogeneous data repositories where query originators have no knowledge of local data organization. A notion of autonomous setting captures the constraints and explicates the *query independence principle*. Two traditional query paradigms – keyword-based search and database framework – are analyzed in this respect. A generalization seamlessly integrating them employs a concept of virtual query objects that provide independent search context and define logical structure of replies. The approach can be practically implemented by extending SQL semantics for the distributed autonomous setting.

Internet search does not require any knowledge of how many websites are relevant or how the data is organized there. Too bad we cannot handle millions of pages often listed in response, most of them irrelevant. Therefore, e.g., we sometimes buy goods at higher prices than we could, since merchants pay search engine companies for the top placement in the search outputs. When querying databases, the responses are precise. Moreover, we get real-time data, not the stuff collected by a crawler a while ago. But we must know the database schema to formulate a query and cannot simultaneously access all databases in the world, even potentially. E.g., can the US government query all its databases at once?

Would it ever be possible to search with one and the same query through all the data of humanity (OK, potentially), like websites, databases, XML repositories, or spreadsheets on the laptops of field agents?

Our solution elaborates on the following *autonomous setting*. 1) The distributed system is comprised of an unfixed number of *data sites*. 2) The sites are completely independent and do not expose any information about their *local data repositories* (models, schemas, attributes, tags, etc.) to the outside world. 3) However, each data site has access to a global information resource - *virtual query objects* (VQOs) - that are used by the data sites to interpret *queries*. 4) The queries are formulated in an SQL-like language using VQOs. 5) As it is clear from the above, the queries cannot be routed to the data sites based on their relevance. All queries are sent to all data sites (theoretically), but not all sites generally answer each query. 6) *Rules of engagement* provide an algorithm to determine if a site can answer a query. 7) *Intersection algorithm* defines the maximal subset of data a site is able to produce in response to a query. In the pure flat relational case, the result is exactly the same a regular SQL query would produce. In the pure keyword case, it works just like traditional keyword-based search. 8) Finally, *security settings* of a site may narrow down the output with respect to the *security context of a query*.

# $\tau$ -xSynopses – a System for Run-Time Management of XML Synopses

Natasha Drukh, Yariv Matia, Yossi Matias, and Leon Portman

School of Computer Science  
Tel Aviv University

kreimern@cs.tau.ac.il, matiayar@post.tau.ac.il, matias@cs.tau.ac.il,  
leon.portman@nice.com

**Abstract.** We introduce the  $\tau$ -xSynopses, a system designed to support remote execution of synopses for XML databases. The  $\tau$ -xSynopses system extends the design and the implementation of the  $\tau$ -Synopses system, enabling a single environment for managing synopses for both XML and relational databases. The system enables easy registration of new synopses from remote platforms, after which the system can manage these synopses, including triggering their construction, rebuilding and update, and invoking them for approximate query processing.

## 1 Introduction

Data synopses are concise representations of data sets, which enable effective processing of approximate queries to the data sets. Recent interest in approximate query processing and in effectively dealing with massive data sets resulted with a proliferation of new synopses.

The  $\tau$ -Synopses system [3] was designed to provide a run-time environment for local and remote execution of various synopses. It provides the management functionality for registered synopses, and it enables easy registration of new synopses either locally or from remote SOAP-enabled platforms. The  $\tau$ -Synopses system can serve as an effective research platform for experimental evaluation and comparison of different synopses, as well as a platform for studying the effective management of multiple synopses in a federated or centralized environment.

The system was previously presented in the context of remote-synopses for relational databases, demonstrating how synopses can be managed in a distributed fashion [2]. It was also presented in the context of synopses management in a single server, and the relevant issues of synopses reconciliation [1].

We introduce the  $\tau$ -xSynopses, a system designed to support remote execution of synopses for XML databases. The  $\tau$ -xSynopses system extends the design and the implementation of the  $\tau$ -Synopses system, enabling a single environment for managing synopses for both XML and relational databases.

## 2 $\tau$ -xSynopses XML Support Specification

The main operational processes supported by the XML support of the  $\tau$ -xSynopses are: constructing of number of pluggable synopses, constructing query workload



and approximate query processing. The system currently supports XML data sources and querying an XML data source with XPath expressions.

The user interface provides an administrator user with a capability to manage data sources, synopsis specifications, updates and pre-defined workloads. The framework provides also a user management mechanism.

- *data sources*: XML data source is specified by an absolute path to a data file and optionally a path for a validating schema file. Relational data source is specified by the database connection information and the target table name plus filter and data column names.
- *synopses*: A synopsis is registered by providing the location of the execution host of the synopsis (server name and the listening port) and the data source for which the synopsis will be built. Invoking the construction of the synopsis requires setting the size limit and optionally the training workload. Both data source and the workload are chosen among those registered in the system.
- *workloads*: The queries workload is produced by the system based on user specifications. The users set the number of queries in the workload, the data relation to be processed and the workload construction parameters. For XML workload the parameters include the minimum and maximum lengths of the path expression and of the branching predicates, as well as the maximum number of branching predicates and the probability of their appearance. For workloads on relational data the parameters are the zipfian skew, low and high values, focus and range values.

The end-user is supplied with the capability to test and compare different synopses that are registered in the system. The Query execution mode provides the user with the ability to evaluate single synopsis at a time. The user input query (an XPath expression for querying XML synopsis or a range query for relational synopsis) or a predefined workload are evaluated against both the synopsis and the real data. The Benchmark mode enables multiple synopses evaluation over pre-defined workloads and their comparison using visual display. The user chooses the synopses to evaluate and the workload to be used for the evaluation. The system calculates the accuracy of the different synopses using several error metrics.

## References

1. Y. Matia, Y. Matias, and L. Portman. Synopses reconciliation via calibration in the  $\tau$ -Synopses system. In *Proc. EDBT 06', Software Demo*, pages 1139–1142, 2006.
2. Y. Matias and L. Portman.  $\tau$ -Synopses: a system for run-time management of remote synopses. In *International Conference on Data Engineering (ICDE), Software Demo*, pages 964–865, April 2004.
3. Y. Matias, L. Portman, and N. Drukh. The design and architecture of the  $\tau$ -Synopses system. In *Proc. EDBT 06', Industrial and Application*, pages 1088–1091, 2006.

# LTS: The List-Traversal Synopses System

Michael Furman<sup>1,\*</sup>, Yossi Matias<sup>1,\*</sup>, and Ely Porat<sup>2</sup>

<sup>1</sup> School of Computer Science, Tel Aviv University

<sup>2</sup> Department of Computer Science, Bar-Ilan University

furman.michael@gmail.com, matias@cs.tau.ac.il,  
porately@cs.biu.ac.il

**Abstract.** The List-Traversal Synopses (LTS) system enables the efficient execution, utilization, and integration of memory-efficient algorithms that can support efficient forward and backward traversal of unidirectional lists. It has applications ranging from traversal of linked structures, garbage collection, hash-chain computations and program rollback. The system provides animation and performance testing platform for traversal algorithms. We will demonstrate the algorithms animation, and will demonstrate how computer programs can be made reversible, and actually run backwards, using the LTS system.

**Keywords:** pebbling, program rollback, hash chain, list traversal, synopsis.

## 1 Introduction

A *list-traversal synopsis* is a memory-efficient data structure that can support efficient forward and backward traversal of unidirectional lists. There is a natural tradeoff between the synopsis size  $\sigma$  and the guaranteed time  $\beta$  per back-step on the list. Obtaining a linear tradeoff,  $\sigma\beta = O(n)$  for a list of size  $n$ , is a simple exercise. We have recently presented [1,2] a family of pebbling-based algorithms that obtain a sub-linear trade-off, for any setting of  $\sigma$  or  $\beta$ , while maintaining a constant time for forward steps. For the setting of  $\sigma = \beta = O(\log n)$ , we showed that using  $O(\log n)$  memory, the  $i$ 'th back-step from the farthest point reached so far takes  $O(\log i)$  time in the worst case, with constant overhead for forward steps. An arbitrary sequence of forward and back steps is allowed.

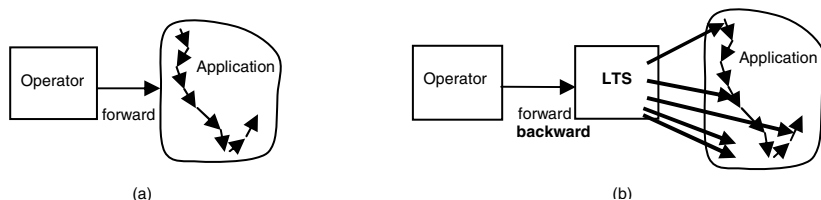
Efficient list-traversal synopses have variety of applications. They can serve to support arbitrary traversals on unidirectional list structures. Two examples are obtaining doubly-linked list functionality out of a unidirectional linked list, and traversing linked structures created in garbage collection algorithms. A unidirectional linked list may also represent a unidirectional process. Indeed another application is that of efficient rollback computation on a hash-chain which is used in various security-related applications. Perhaps the most interesting application is in supporting program rollback for arbitrary programs. That is, enable any non-interactive computer program be reversible with little memory and marginal time overheads.

---

\* Supported in part by grants from the Israel Science Foundation.

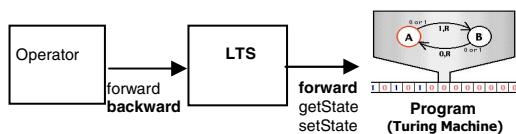
## 2 The List-Traversal Synopses (LTS) System

We have developed the List Traversal Synopses system, which has the following features: (i) it can accommodate *multiple pebbling algorithms* and various *list-traversal synopses* implementations; (ii) it enables effective *animation* of the various synopses and algorithms; (iii) it has an infrastructure of effective *performance testing*; (iv) it has a simple API that allows integrating it with various applications, as depicted in Fig. 1. The system includes implementations and animations of host of pebbling algorithms – from the basic ones to the most advanced ones. It allows applying them on various applications, including list traversals, hash chain utilization, and the backward execution of computer simulations.



**Fig. 1.** An original application (a) and a modified application (b) enabled with the LTS system. (a) A computer program (operator) uses solely forward steps in a unidirectional process. (b) The operator can perform back traversal on the application using the LTS system.

We will demonstrate the algorithms animation, their performance testing, and we will demonstrate the application for effectively reversing a computer program, as depicted in Fig 2. More details about the system, its architecture, and implementation are available at [3] and at <http://www.cs.tau.ac.il/~matias/lts/>



**Fig. 2. Program Rollback.** LTS allows rollback of any application using  $\log n$  program states in  $O(\log i)$  time per the  $i$ 'th back step. To allow LTS to bind to the application, it is required to identify the application state. The state should contain complete save / restore information.

## References

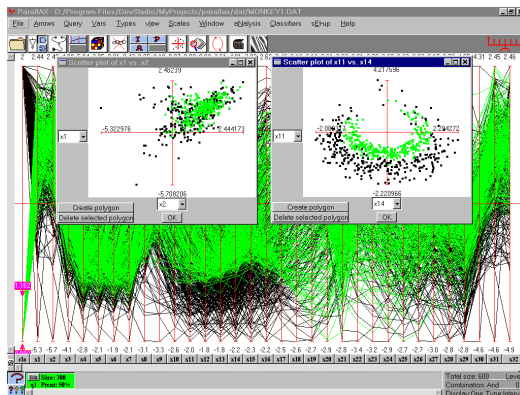
1. Matias, Y., Porat, E.: Efficient pebbling for list traversal synopses, In Thirtieth International Colloquium on Automata, Languages and Programming (ICALP), (2003) 918-928.
2. Matias, Y., Porat, E.: Efficient pebbling for list traversal synopses with applications to program rollback. Theory of Computing Systems (TOCS), to appear.
3. Furman, M., Matias, Y., Porat, E.: The List Traversal Synopses (LTS) System. Technical Report, Tel Aviv University (2006).

# The Promise and Challenge of Multidimensional Visualization

Alfred Inselberg\*

School of Mathematical Sciences  
Tel Aviv University  
Tel Aviv, Israel  
aiisreal@math.tau.ac.il  
www.math.tau.ac.il/~aiisreal

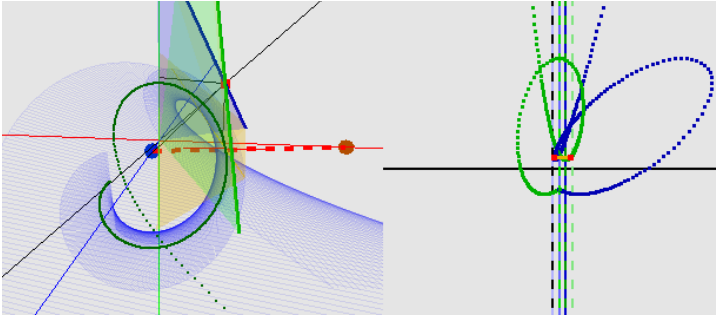
A dataset with  $M$  items has  $2^M$  subsets anyone of which may be the one we really want. With a good data display our fantastic pattern-recognition ability can not only cut great swaths searching through this combinatorial explosion but also extract insights from the visual patterns. These are the core reasons for data visualization. With Parallel Coordinates (abbr.  $\parallel$ -coords) the search for multivariate relations in high dimensional datasets is transformed into a 2-D pattern recognition problem. After a short overview of  $\parallel$ -coords, guidelines and strategies for knowledge discovery are illustrated on different real datasets, one with 400 variables from a manufacturing process. A geometric classification algorithm based on  $\parallel$ -coords is presented and applied to complex datasets. It has low computational complexity providing the classification rule explicitly and



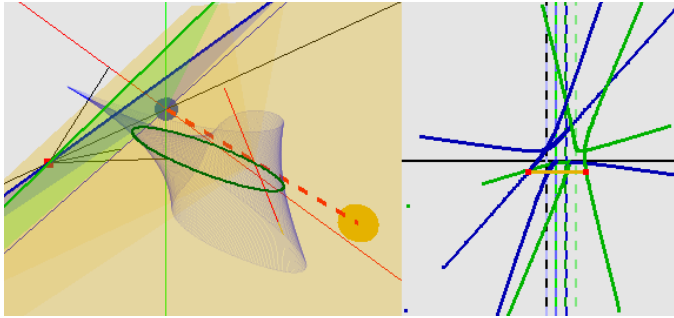
**Fig. 1.** Dataset with 32 variables with 2 categories (2 colors). First two variables are plotted on the left. The classifier found, within a 4 % error, that one of the categories is 9-dimensional ordering its variables by their predictive value; the top two are plotted (right) showing the separation achieved.

---

\* Senior Fellow San Diego SuperComputing Center, San Diego, & Multidimensional Graphs Ltd, Raanana 43556, Israel.



**Fig. 2.** A winding helicoid in 3-D in cartesian and  $\parallel$ -coords. Two intersecting lines (left) specify one of the helicoid's tangent planes represented by a pair of points one on each of the curves (right). A helicoid in  $N$ -D is represented by  $N - 1$  such curves (right).



**Fig. 3.** Path on a Moebius strip (*non-orientable surface*) and one of its tangent planes. In  $N$ -D it is represented by  $N - 1$  such curves.

*visually*. The minimal set of variables required to state the rule is found and ordered by their predictive value (see Fig. 1). A visual economic model of a real country is constructed and analyzed to illustrate how multivariate relations can be modeled by means of hypersurfaces, understanding trade-offs, sensitivities and interrelations. Examples of complex hypersurfaces which can be represented in  $\parallel$ -coords are shown in Figs 2 and 3. Parallel coordinates is also used in collision avoidance algorithms for air traffic control.

P.S. Do not be intimidated by this formalistic description. The speaker is also well known for his numerological anecdotes and palindromic digressions!

# *OPOSSUM*: Bridging the Gap Between Web Services and the Semantic Web

Eran Toch<sup>1</sup>, Iris Reinhartz-Berger<sup>2</sup>, Avigdor Gal<sup>1</sup>, and Dov Dori<sup>1</sup>

<sup>1</sup> Technion – Israel Institute of Technology, Technion City, Haifa 32000, Israel  
erant@tx.technion.ac.il, {avigal, dori}@ie.technion.ac.il

<sup>2</sup> University of Haifa, Carmel Mt., Haifa 31905, Israel  
iris@mis.haifa.ac.il

## 1 Introduction and Background

Web services are distributed software components, which are accessed through the World Wide Web. The increasing use of Web services raises the need for efficient and precise retrieval solutions of Web services. We propose to investigate aspects of Web service retrieval, facing a gap between user specifications, given in some form of a semantic description, and Web service definition, given in a standard interface description such as Web Service Description Language (WSDL), that conveys the syntax of the service. Bridging this gap is becoming more and more urgent as users need to find Web services among increasing numbers of Web services within organizations and on the Web. One of the major challenges of service-retrieval is to make services accessible without having to do additional semantic work of classification.

The main challenge in service-retrieval is the lack of semantics in their interface description for precise search. Current retrieval solutions can be classified as either *linguistic-based* or *semantics-based*, each of which exhibits different limitations. Linguistic-based solutions are based on textual analysis of WSDL description. This approach is taken, for example, by the Woogole search engine [3]. While linguistic approaches dramatically increase the precision and recall of finding WSDL-described Web services, they do not reach the level of certainty required to support a fully automated solution. The semantic approach to service-retrieval is based on expanding the description of Web services with formal semantic models, such as OWL-S [1]. These models provide an unambiguous description of service properties by relating them to concepts belonging to *Web ontologies*. Several works, including [4], had proposed methods for matching queries and advertisements of semantic Web services. While these methods retrieve services with very high precision and certainty, they require full semantic models in order to perform the matching. These semantic models may not be available in all application fields.

## 2 Ontological-Based Search

In order to address the current limitations of service-retrieval, we had developed *OPOSSUM* (Object-Procedure-SemanticS Unified Matching). Our approach

relies on ontologies as a mean for increasing certainty of matching service functionality. By using data integration and conceptual model techniques, WSDL documents are analyzed and transformed into a generic service model. Following that, the service properties are mapped to concepts that belong to ontologies, which are collected from the Semantic Web [2]. For instance, a corporate book-buying service will be mapped to ontologies that describe e-commerce, finance, and corporate concepts.

The service-retrieval problem is transformed from a mapping problem between a query and a set of services to a mapping problem between a set of query concepts and a set of service concepts. A novel algorithm [5], is used to match queries with the ontological-based model of Web services. Rather than using purely linguistic methods for measuring the affinity between concepts, the matching algorithm analyzes the ontology, using related concepts and the structure of the ontology itself. Results are ranked according to the accumulated certainty (over the set of query expressions, concepts and messages) of the mapping between the query and the services. Similarly, semantic affinity methods are used to determine relations between Web service operations. These relations are used in order to support retrieval of composed services.

Users communicate with the *OPOSSUM* search engine using a simple query language, which is based on a natural language. A set of query operators enable users to refer to specific properties of desired Web services. For instance, the operator *input* returns all services that have an input that correlates with the query concept. Other operators include *output*, *description*, *name*, *etc*, as well as disjunction and conjunction of properties. Furthermore, the search engine can retrieve dynamically-created services, which are composed of several atomic services. Thus, even if a query is not answered by a single service, it might be answered by combining several services. Compositions of Web services are evaluated in a preprocessing stage, in which semantic methods are used in order to infer the underlying relations between individual Web services.

## References

1. Anupriya Ankolekar, David L. Martin, Zeng Zeng, Jerry R. Hobbs, Katia Sycara, Burstein Burstein, Massimo Paolucci, Ora Lassila, Sheila A. McIlraith, Srini Narayanan, and Payne Payne. DAML-S: Semantic markup for web services. In *Proceedings of the International Semantic Web Workshop (SWWS)*, pages 411–430, July 13 2001.
2. T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, 2001.
3. Xin Dong, Alon Y. Halevy, Jayant Madhavan, Ema Nemes, and Jun Zhang. Similarity search for web services. In *VLDB*, pages 372–383, 2004.
4. Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, and Katia P. Sycara. Semantic matching of web services capabilities. In *International Semantic Web Conference*, pages 333–347, 2002.
5. Eran Toch, Avigdor Gal, and Dov Dori. Automatically grounding semantically-enriched conceptual models to concrete web services. In *ER*, pages 304–319, 2005.

# *ProMo* - A Scalable and Efficient Framework for Online Data Delivery

Haggai Roitman<sup>1</sup>, Avigdor Gal<sup>1</sup>, and Louiqa Raschid<sup>2</sup>

<sup>1</sup> Technion - Israel Institute of Technology  
Haifa 32000, Israel

{haggair@tx, avigal@ie}.technion.ac.il

<sup>2</sup> University of Maryland, College Park  
MD 20742 USA

louiqa@umiacs.umd.edu

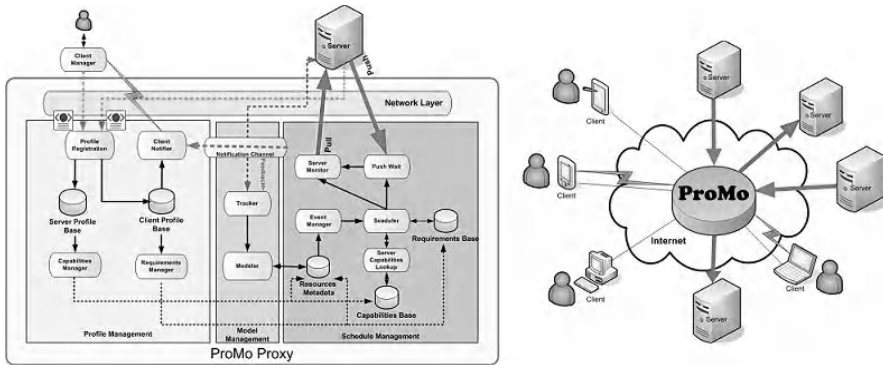
## 1 Introduction

Web enabled application servers have had to increase the sophistication of their server capabilities in order to keep up with the increasing demand for client customization. Typical applications include RSS feeds, stock prices and auctions on the commercial Internet, and increasingly, the availability of Grid computational resources. Web data delivery technology has not kept up with these demands. There still remains a fundamental trade-off between the scalability of both performance and ease of implementation on the server side, with respect to the multitude and diversity of clients, and the required customization to deliver the right service/data to the client at the desired time. Current data delivery solutions can be classified as either push or pull solutions, each suffering from different drawbacks. Push is not scalable, and reaching a large numbers of potentially transient clients is typically expensive in terms of resource consumption and implementation by a server. In some cases, where there is a mismatch with client needs, pushing information may overwhelm the client with unsolicited information. Pull, on the other hand, can increase network and server workload and often cannot meet client needs. Several hybrid push-pull solutions have also been presented in the past. In this demonstration we present *ProMo*, a scalable and efficient hybrid data delivery solution.

## 2 *ProMo*

*ProMo* is a framework that includes a language, model, and algorithms to support flexible, efficient and scalable targeted data delivery. It is flexible since it can accommodate push, pull or hybrid push-pull solutions. It is efficient since it exploits server capabilities in meeting client demands. *ProMo* provides a uniform specification language to define expressive profiles that can be used by clients to specify client data needs and that can be used by servers to specify server data delivery capabilities. The profiles are then translated into *execution intervals*, an abstraction presented in [1] that identifies periods of time in which client needs should be met and server capabilities can be exploited to meet





**Fig. 1.** *ProMo* Framework

those needs. Client profiles can be further aggregated and server profiles can be composed to build efficient push-pull schedules. Within this framework, a proxy for the client takes the responsibility to match client needs with relevant server capabilities. The proxy determines if the server’s capabilities can satisfy (or partially satisfy) the client’s needs. Further, the proxy generates a data delivery schedule that exploits the available server capabilities. To do so, the proxy will exploit server push, and as needed, it may augment server push with pull actions.

### 3 Demonstration Details

Using the GUI of the demo, we shall specify different client and server profiles for several RSS ([2]) feeds taken from CNN ([3]). For demonstration purposes we assume that CNN server support an extension of RSS 2.0 `cclouds` (refer to RSS 2.0 specification in [2]) and can push the proxy the set of items that have been changed according to the server profile. For a chosen profile and updates at the server, we illustrate the *ProMo* schedule and show how *ProMo* will exploit push from the server and augment with pull actions. We shall further use different efficient pull scheduling algorithms suggested in [1] and show that they guarantee to satisfy the client profiles. The implementation is in Java JDK 1.4.2.

## References

1. H. Roitman, A. Gai, L. Bright and L. Raschid: A Dual Framework and Algorithms for Targeted Data Delivery, Technical Report, University of Maryland, College Park. (2005) Available from <http://hdl.handle.net/1903/3012>
2. RSS Specification: <http://www.rss-specifications.com>
3. CNN RSS Services: <http://rss.cnn.com/services/rss/>

# Next Generation Enterprise IT Systems

Donald F. Ferguson

IBM

**Abstract.** The next several years will see an unprecedented change in the architecture of enterprise IT systems. There have been many such changes in the past, for example client-server, user workstations, the rise of Linux-UNIX-Windows transaction and query processing and data mining. The coming wave of change will be at least as great as prior sea changes, and perhaps larger than all combined.

The keynote discussed the changes, and their technical implications for the design of systems, middleware and applications. First, the talk set background and context that enables the massive change. Some elements included:

- The increased adoption of highly parallel systems built using multi-core chips, multiple HW thread cores and blade systems.
- General purpose server HW forming the basis for storage and communication subsystems.
- The increasing rise of standards, especially general Web service standards and domain specific standards.
- The “everyone can program phenomenon”, or all entrants to the work force have basic programming skills.

Typical reference architectures for enterprise IT environments will include a runtime architectures and the development model. The presentation also described the four core approaches to application integration. We then focused on selecting next generation solution elements. Some examples included:

- Situational, ad hoc applications.
- Composite applications
- Patterns, template and recipes
- Reflecting business componentization in SOA realizations
- Integration of non-traditional services into composite applications, including optimization and grid services.
- The convergence of event driven architecture and SOA.

Our industry is at the very beginning of these transitions. Any change of this magnitude creates many new areas for innovation and research. We enumerated some opportunities.

# The Future of Web Search: From Information Retrieval to Information Supply

Andrei Broder

Yahoo! Research

**Abstract.** In the past decade, Web search engines have evolved from a first generation based on classic Information Retrieval (IR) algorithms scaled to web size and thus supporting only informational queries, to a second generation supporting navigational queries using web specific information (primarily link analysis), to a third generation enabling transactional and other "semantic" queries based on a variety of technologies aimed to directly satisfy the unexpressed "user intent."

What is coming next? In this talk, we argue for the trend towards context driven Information Supply (IS), that is, the goal of Web IR will widen to include the supply of relevant information from multiple sources without requiring the user to make an explicit query. The information supply concept greatly precedes information retrieval. What is new in the web framework is the ability to supply relevant information specific to a given activity and a given user, while the activity is being performed. A prime example is the matching of ads to content being read, however the information supply paradigm is starting to appear in other contexts such as social networks, e-commerce, browsers, e-mail, and others.

# Is There Life After the Internet?

Yechiam Yemini

Columbia University

**Abstract.** Recent advances in infrastructure technologies have been radically transforming the fundamentals of computing paradigms. Three concomitant disruptive changes are of particular importance:

- (a) Emerging wire-speeds have been inverting the classical speed hierarchy; with I/O rates outstripping CPU speeds, the focus of traditional architectures, scaling I/O-memory speeds to meet CPU speeds, is inverted to scaling CPU speeds to meet I/O-storage speeds;
- (b) Tagged XML software is inverting the classical sequential-code / random-access-data computing into sequential-data / random-access-code; and
- (c) Massive storage growth at clients and respective download-&-play applications are beginning to stretch the Internet's unicast-client / server-content-distribution-&-processing paradigm to its limits;

Similar disruptive changes have historically transformed mainframe-computing to desktop-computing then to Internet-computing.

It is thus not unreasonable to ask, whether the Internet-computing paradigm may be similarly transformed, and if so, what might be the characteristics of post-Internet computing.

# Author Index

- Álvarez, Manuel 1  
 Asbeh, Nuaman 333  
  
 Balaban, Mira 59, 71  
 Bar-Ilan, Judit 26  
 Barone, Daniele 47  
 Baruchson-Arbib, Shifra 26  
 Batini, Carlo 47  
 Ben-Chaim, Yochai 221  
 Berlin, Jacob 13  
 Berman, Tamar 165  
 Broder, Andrei 362  
  
 Chae, Heung Seok 83  
 Chang, Jae-Woo 249  
 Cheng, Eric Y. 210  
 Cohen, Sara 153  
  
 D'Atri, Alessandro 260  
 Domingo-Ferrer, Josep 106  
 Dori, Dov 186, 357  
 Draheim, Dirk 274  
 Drukh, Natasha 351  
  
 Etzion, Opher 174  
  
 Ferguson, Donald F. 361  
 Frieder, Gideon 38  
 Frieder, Ophir 38  
 Furman, Michael 353  
  
 Gafni, Eli 260  
 Gal, Avigdor 174, 221, 357, 359  
 Gao, Ji 312  
 Gilula, Mikhail 350  
 Gold, Alex 337  
 Grossman, David 38  
 Guerrero, Jaime 1  
 Guo, Hang 312  
  
 Ha, Sungho 83  
 Heard, Jefferson 38  
 Heilili, Nuermaimaiti 96  
 Hidalgo, Justo 1  
 Huang, Jin 198  
  
 Inselberg, Alfred 355  
  
 Jian, Shuo Hung 210  
 Jiang, Lizheng 287  
 Jin, Hai 198  
  
 Kane, Larry 38  
 Kim, Byung Ryong 117  
 Kim, In-Cheol 341  
 Kim, Ki Chang 117  
 Kimelfeld, Benny 141  
 Kuflik, Tsvi 333  
  
 Lee, Jeonghyun 321  
 Lee, Joon-Sang 83  
 Lee, Seungkeun 321  
 Limonad, Lior 59  
 Lin, Zuoquan 96  
 Luo, Zhenxing 96  
  
 Ma, Xiuli 287  
 Maimon, Oded 300  
 Maraee, Azzam 71  
 Matias, Yossi 351, 353  
 Matia, Yariv 351  
 Motro, Amihai 13, 260  
  
 Naggar, Paolo 47  
 No, Jaechun 237  
  
 Pan, Alberto 1  
 Park, Chang Won 237  
 Park, Jaegel 83  
 Park, Jinwook 83  
 Park, Sung Soon 237  
 Peleg, Mor 333  
 Porat, Ely 353  
 Portman, Leon 351  
  
 Rafaeli, Sheizaf 26  
 Raschid, Louiqa 359  
 Ravid, Gilad 26  
 Reinhartz-Berger, Iris 186, 337, 357  
 Roitman, Haggai 359  
 Rokach, Lior 300  
 Romano, Roni 300  
 Ronen, Royi 129

Sagiv, Yehoshua 141  
Schertz, Mitchell 333  
Shlezinger, Galia 186  
Shmueli, Oded 129

Tan, Shaohua 287  
Tang, Shiwei 287  
Thiemann, Peter 274  
Toch, Eran 357

Viscusi, Gianluigi 47

Wang, Tae-Woong 249  
Wasserkrug, Segev 174

Weber, Gerald 274  
Wilberding, Jordan 38

Xie, Xia 198  
Xu, Dawei 96

Yang, Dongqing 287  
Yemini, Yechiam 363  
Yoo, Kee-Young 345  
Yoon, Eun-Jun 345

Zeng, Zhi-Qiang 312  
Zhang, Dehui 287  
Zhang, Qin 198  
Zhao, Chen 96